21世纪高等学校计算机
专业实用规划教材

数据库技术及安全教程
—— SQL Server 2008

◎ 尹志宇 郭晴 主编




数据库技术基础 · 数据库安全 · 数据库开发

清華大學出版社

21世纪高等学校计算机专业实用规划教材

数据库技术及安全教程
——SQL Server 2008

尹志宇　郭　晴　主编
李青茹　解春燕　于富强　陈敬利　副主编

清华大学出版社
北　京

内 容 简 介

本书全面讲述了数据库的基本概念、安全和 SQL Server 2008 的技术和应用,全书以理论够用、实践第一为原则,使读者能够快速、轻松地掌握 SQL Server 数据库技术与安全相关知识。全书分三部分:第一部分讲述 SQL Server 2008 的数据库技术基础,内容包括 SQL Server 2008 概述、数据库和表的基本概念和操作、数据库查询、T-SQL 编程、视图和索引、存储过程和触发器、事务和并发控制等;第二部分讲述数据库安全,内容包括数据库访问的安全性管理、数据加密和攻击防范、数据库文件的安全备份与恢复以及数据库复制;第三部分讲述数据库开发,内容包括数据库设计和基于 C#.NET 的 SQL Server 数据库应用系统开发过程。

本书理论和实践相结合,既阐述了数据库的基本概念和安全理论,又结合了 SQL Server 2008 数据库管理系统的应用。本书内容翔实,实例丰富,图文并茂,体系完整,通俗易懂,有助于读者理解数据库的基本概念,掌握要点和攻克难点;为便于学习,每章还配有丰富的习题。

本书可作为大学本科、专科、高等职业院校"SQL Server 数据库技术及安全管理"相关课程的教学用书,也可以作为培养数据库系统工程师的培训教材,还可以作为数据库管理人员及数据库应用系统开发人员的参考用书。

图书在版编目(CIP)数据

数据库技术及安全教程:SQL Server 2008/尹志宇等主编. —北京:清华大学出版社,2018
(21 世纪高等学校计算机专业实用规划教材)
ISBN 978-7-302-49501-7

Ⅰ. ①数… Ⅱ. ①尹… Ⅲ. ①关系数据库系统—安全管理—教材 Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字(2018)第 020799 号

责任编辑:魏江江 李 晔
封面设计:刘 键
责任校对:时翠兰
责任印制:丛怀宇

出版发行:清华大学出版社
网 址:http://www.tup.com.cn,http://www.wqbook.com
地 址:北京清华大学学研大厦 A 座 **邮 编**:100084
社 总 机:010-62770175 **邮 购**:010-62786544
投稿与读者服务:010-62776969,c-service@tup.tsinghua.edu.cn
质量反馈:010-62772015,zhiliang@tup.tsinghua.edu.cn
课件下载:http://www.tup.com.cn,010-62795954
印 刷 者:北京富博印刷有限公司
装 订 者:北京市密云县京文制本装订厂
经 销:全国新华书店
开 本:185mm×260mm **印 张**:23.75 **字 数**:571 千字
版 次:2018 年 9 月第 1 版 **印 次**:2018 年 9 月第1 次印刷
印 张:1～1500
定 价:59.50 元

产品编号:076317-01

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施“高等学校本科教学质量与教学改革工程（简称‘质量工程’）”，通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合“质量工程”的实施，满足各高校教学质量和教学改革的需要。

本系列教材立足于计算机专业课程领域，以专业基础课为主、专业课为辅，横向满足高校多层次教学的需要。在规划过程中体现了如下一些基本原则和特点。

（1）反映计算机学科的最新发展，总结近年来计算机专业教学的最新成果。内容先进，充分吸收国外先进成果和理念。

（2）反映教学需要，促进教学发展。教材要适应多样化的教学需要，正确把握教学内容和课程体系的改革方向，融合先进的教学思想、方法和手段，体现科学性、先进性和系统性，强调对学生实践能力的培养，为学生知识、能力、素质协调发展创造条件。

（3）实施精品战略，突出重点，保证质量。规划教材把重点放在公共基础课和专业基础课的教材建设上；特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现教学质量和教学改革成果的教材。

（4）主张一纲多本，合理配套。专业基础课和专业课教材配套，同一门课程有针对不同层次、面向不同应用的多本具有各自内容特点的教材。处理好教材统一性与多样化，基本教材与辅助教材、教学参考书，文字教材与软件教材的关系，实现教材系列资源配套。

（5）依靠专家，择优选用。在制定教材规划时要依靠各课程专家在调查研究本课程教

材建设现状的基础上提出规划选题。在落实主编人选时，要引入竞争机制，通过申报、评审确定主题。书稿完成后要认真实行审稿程序，确保出书质量。

繁荣教材出版事业，提高教材质量的关键是教师。建立一支高水平教材编写梯队才能保证教材的编写质量和建设力度，希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21世纪高等学校计算机专业实用规划教材

联系人：魏江江 weijj@tup.tsinghua.edu.cn



前　言

数据库最初是在大公司或大机构中用作大规模事务处理的基础。后来随着个人计算机的普及，数据库技术被移植到 PC 上，供单用户个人数据库应用。接着，由于 PC 在工作组内连成网，数据库技术就移植到工作组级。如今，数据库正在 Internet 和内联网中广泛使用。

数据库技术是目前计算机领域发展最快、应用最广泛的技术，它的应用遍及各行各业，大到操作系统程序，如全国联网的飞机票、火车票订票系统，银行业务系统，小到个人的管理信息系统，如家庭理财系统。随着数据库应用越来越广泛，数据库的安全问题也越来越重要，数据泄露、数据被破坏、网站被攻击等案例层出不穷，在互联网日渐流行的动态网站中，数据库的应用及安全管理显得尤为重要。

SQL Server 2008 是一个功能完备的数据库管理系统，提供了完整的关系数据库创建、开发、安全和管理功能。它功能强大、操作简便，日益被广大数据库用户所喜爱，而且越来越多的开发工具都提供了与 SQL Server 的接口。

本书系统地介绍数据库技术的基本理论，全面介绍 SQL Server 2008 的各项功能、数据库系统设计方法、安全及管理以及数据库系统开发应用的相关技术。全书共 14 章，分为三个部分：第一部分是第 1～8 章，系统讲述数据库的基本概念和技术，其内容包括数据库系统简介和 SQL Server 2008 基础、数据库基本概念和操作、表的操作、数据库查询、T-SQL 编程、视图和索引、存储过程和触发器、事务和并发控制；第二部分是第 9～12 章，讲述数据库的安全性管理，其内容包括数据库访问的安全性管理、数据加密和攻击防范、数据库文件的安全备份与恢复以及数据库复制；第三部分是第 13 章和第 14 章，首先讲述数据库设计详细过程，然后利用一个“教学管理系统”实例介绍了基于 C#.NET 的 SQL Server 数据库应用系统的开发过程。

本书作者长期从事本科计算机类专业的教学工作，不仅具有丰富的教学经验，同时还具有多年的数据库开发经验，深知数据库的主要知识点和重点、难点，了解什么样的教材适合教学使用，知道学生及各类读者对数据库的学习方式和兴趣所在，以及如何组织书本的内容更利于教学和自学，从而形成本书的结构体系。

本书内容翔实、体系完整、图文并茂，具有较强的系统性和实用性。章节安排合理，理论与实践紧密结合，每章后面都配有习题，有助于读者理解概念、巩固知识、掌握重点、攻克难点。

本书既可作为本科院校计算机应用专业、计算机信息管理专业及相关专业的教材和参考书，也适合数据库管理人员及数据库应用系统开发人员参考，还可以作为计算机爱好者的自学参考书。

本书第 6～8 章和第 11 章由尹志宇编写，第 1～4 章由郭晴编写，第 12 章由陈敬利编

写，第 9 章和第 10 章由李青茹编写，第 5 章由解春燕编写，第 13 章由于富强编写，第 14 章由解春燕和李涵玥共同完成。全书由尹志宇统稿。

为了配合教学和参考，本书提供了配套的电子教案，读者可到清华大学出版社网站(http://www.tup.com.cn)下载。

由于编者水平有限，书中难免有疏漏与错误之处，衷心希望广大读者批评、指正。

编　者

2018 年 3 月



第一部分　SQL Server 2008 的数据库技术基础

第三部分 数据库开发

第一部分

SQL Server 2008 的数据库技术基础

第 1 章　SQL Server 2008 概述

数据库技术是一门信息管理自动化学科，是计算机学科的一个重要分支。数据库技术所研究的问题是如何科学地组织和存储数据，在数据库系统中减少数据存储冗余、实现数据共享，以及如何保障数据安全、有效地获取和处理数据。

本章简要介绍数据库系统基础，包括数据库技术的发展历史、数据库系统的基本组成以及数据库的数据模型表示；SQL Server 2008 基础，包括 SQL Server 发展史、SQL Server 2008 新增功能和性能、SQL Server 2008 的版本和安装、SQL Server 2008 的基本组成介绍。

1.1　数据库系统简介

从 20 世纪 60 年代末到现在，数据库技术已经发展了 50 多年。在这 50 多年的历程中，人们在数据库技术的理论研究和系统开发上取得了辉煌的成就，数据库系统已经成为现代计算机系统的重要组成部分。数据库技术最初是在大公司或大机构中用于大规模事务处理；随着个人计算机的普及，数据库技术被移植到 PC 上供单用户应用；接着，由于 PC 在工作组内连成网，数据库技术就移植到工作组级；如今，数据库技术正在 Internet 中被广泛使用。

1.1.1　数据库技术发展史

数据(Data)是描述现实世界中各种具体事物或抽象概念的符号记录。除了常用的数字数据外，文字(如名称)、图形、图像、声音等信息，也都是数据。在日常生活中，人们使用交流语言(如汉语)去描述事物；在计算机中，为了存储和处理这些事物，就要抽出这些事物的特征组成一个记录来描述事物。例如，在学生管理中，可以对学生的学号、姓名、性别和年龄等情况这样描述：201601001，张三，男，19。

数据处理(Data Process)是指对数据的收集、分类、组织、编码、存储、加工、计算、检索、维护、传播以及打印等一系列的活动。数据处理的目的是从大量的数据中，根据数据自身的规律及其固有的联系，通过分析、归纳、推理等科学手段，提取出有效的信息资源。

在数据处理中，通常数据的加工、计算等比较简单，而数据的管理比较复杂。数据管理是数据处理的核心，是指数据的收集、分类、组织、编码、存储、检索、维护等操作，这部分操作是数据处理业务的基本环节，是任何数据处理业务中必不可少的共有部分，因此学习和掌握数据管理技术，能对数据处理提供有力的支持。

随着计算机硬件和软件的发展，数据库技术也不断地发展。从数据管理的角度，数据库技术经历了人工管理、文件系统和数据库系统三个阶段。

1. 人工管理阶段

20 世纪 50 年代中期以前，计算机主要用于科学计算。从硬件上看，外部存储器只有磁带、卡片和纸带等，还没有磁盘等直接存取存储设备；软件没有操作系统，没有管理数据的软件，没有高级语言，只有汇编语言。

这个时期数据管理的特点如下。

(1) 数据无独立性。数据由计算或处理它的程序自行携带，程序和数据是一个不可分割的整体，应用程序依赖于数据的物理组织，数据脱离了程序就无任何价值。数据只有与相应的程序一起保存才有价值，否则就毫无用处。所以，所有程序的数据均不能单独保存。

(2) 数据不能共享。不同的程序均有各自的数据，这些数据对不同的程序通常是不相同的，当然不可共享；即使不同的程序使用了相同的一组数据，这些数据也不能共享，程序中仍然需要各自加入这组数据，谁也不能省略。这种数据的不可共享性，必然导致程序与程序之间存在大量的重复数据，浪费了存储空间。

人工管理阶段应用程序与数据之间的关系如图 1-1 所示。

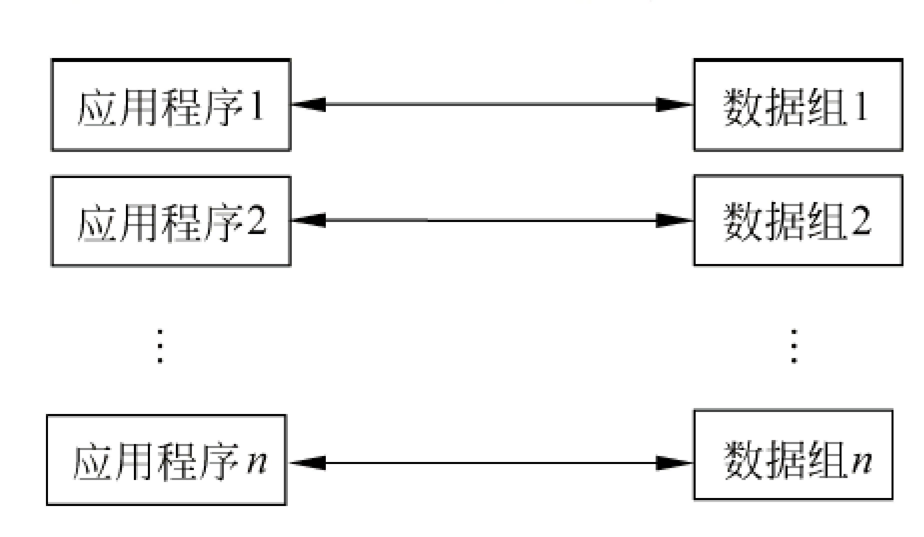


图 1-1　人工管理阶段应用程序与数据的对应关系

2. 文件系统阶段

20 世纪 50 年代后期至 60 年代中后期，计算机的应用范围逐渐扩大，不仅用于科学计算，还大量用于管理。随着数据量的增加，数据的存储、检索和维护问题成为紧迫的需要，数据结构和数据管理技术迅速发展起来。硬件方面，外部存储器已有磁盘、磁鼓等直接存取的存储设备；软件方面，出现了高级语言和操作系统。操作系统中的文件系统是专门管理外存的数据管理软件，该系统把计算机中的数据组织成相互独立的数据文件。数据处理方式有批处理，也有联机实时处理。

这个阶段有如下几个特点。

(1) 数据以“文件”形式可长期保存在外部存储器的磁盘上。对数据的操作以记录为单位，文件的建立、存取、查询、插入、删除、修改等所有操作，都要用程序来实现。

(2) 程序与数据之间具有了一定的独立性——“设备独立性”，即程序只需用文件名就可与数据打交道，不必关心数据的物理位置。

(3) 文件结构的设计仍然是基于特定的用途，程序基于特定的物理结构和存取方法，因此程序与数据结构之间的依赖关系并未根本改变。由于文件之间缺乏联系，造成每个应用程序都有对应的文件，同样的数据有可能在多个文件中重复存储，数据冗余度大。

文件系统阶段应用程序与数据之间的关系如图 1-2 所示。

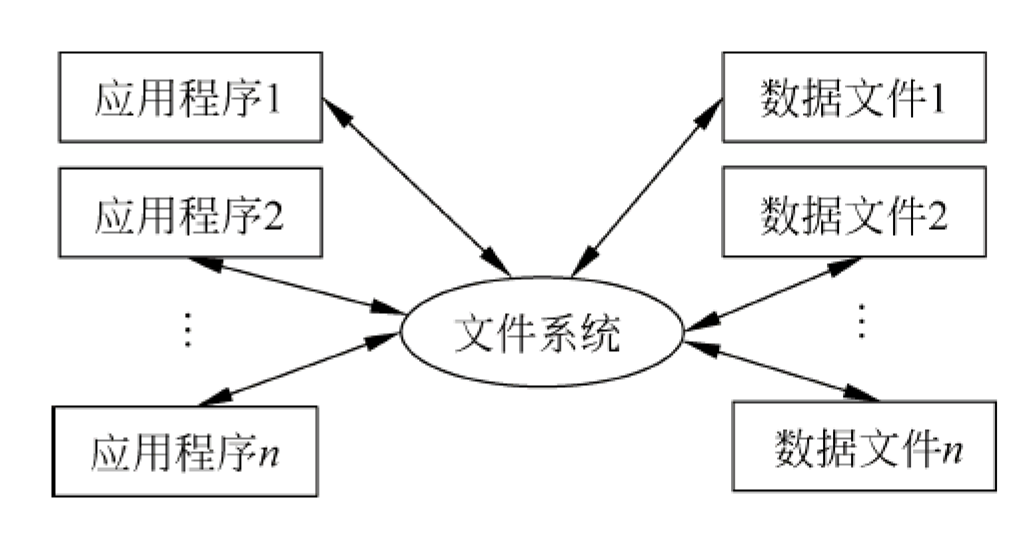


图 1-2　文件系统阶段应用程序与数据的对应关系

3. 数据库系统阶段

20 世纪 60 年代后期以来，计算机应用越来越广泛，数据量急剧增加，而且数据的共享要求越来越高。计算机的硬件和软件都有了进一

步的发展，硬件方面，有了大容量的磁盘；软件方面，传统的文件系统已经不能满足人们的需求，能够统一管理和共享数据的数据库管理系统(Database Management System，DBMS)应运而生。所以，此阶段将数据集中存储在各数据库中，由 DBMS 进行统一组织和管理。从处理方式上讲，联机实时处理要求更多了，并开始提出和考虑分布处理。

数据库系统阶段的特点也是优点如下。

1) 数据结构化

数据结构化是数据库系统与文件系统的根本区别。有了 DBMS 后，数据库中的数据不再针对某一应用，而是面向整个应用系统，它是对整个组织的各种应用(包括将来可能的应用)进行通盘考虑后建立起来的总的数据结构。

2) 较高的数据共享性

数据共享是指允许多个用户同时存取数据而互不影响，该特征正是数据库技术先进性的体现。数据库系统从整体角度描述数据，数据不再面向某个应用而是面向整个系统，因此数据可以被多个用户、多个应用共享使用。数据共享可以大大减少数据冗余，节约存储空间。

3) 较高的数据独立性

所谓数据独立，是指数据与应用程序之间的彼此独立，它们之间不存在相互依赖的关系。应用程序不随数据存储结构的变化而变化，因为应用程序以简单的逻辑结构操作数据而无须考虑数据的物理结构，简化了应用程序的编制和程序员的工作负担。

4) 数据由 DBMS 统一管理和控制

数据库的共享是并发的共享，即多个用户可以同时存取数据库中的数据，甚至可以同时存取数据库中的同一数据。因此，DBMS 还必须提供数据的统一管理和控制功能。

DBMS 加入了安全保密机制，可以防止数据被非法存取；DBMS 的数据完整性保护可以保障数据的正确性、有效性和相容性，完整性检查将数据控制在有效的范围内或保证数据之间满足一定的关系；当多个用户的并发进程同时存取、修改数据库时，可能会发生相互干扰而得到错误的结果，或使得数据库的完整性遭到破坏，因此 DBMS 必须对多用户的并发操作加以控制和协调；另外，DBMS 还采取了一系列措施，以实现对数据库破坏后的恢复。

数据库系统阶段应用程序与数据之间的关系如图 1-3 所示。

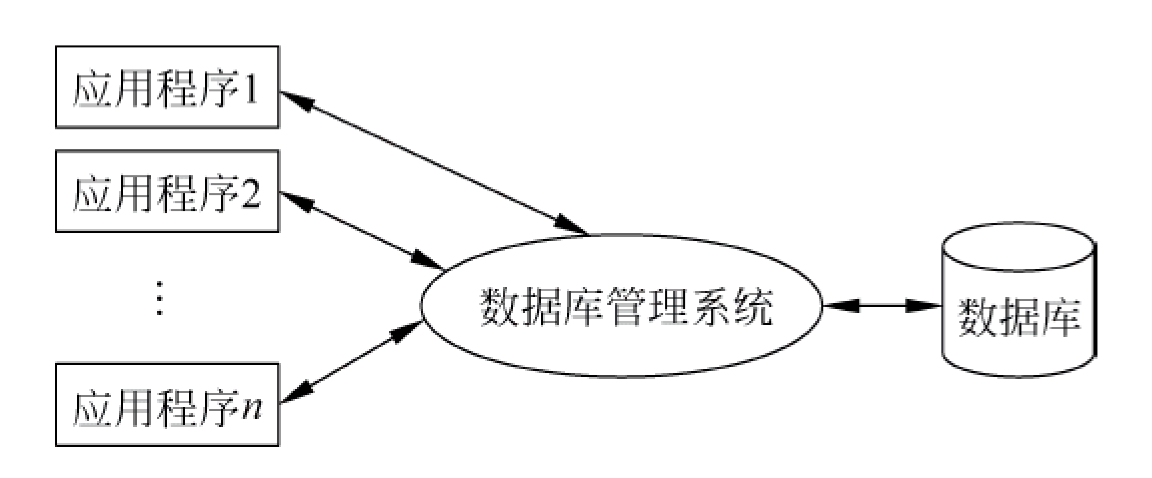


图 1-3 数据库系统阶段应用程序与数据的对应关系

1.1.2 数据库系统组成

数据库系统(Database System，DBS)是指在计算机系统中引入数据库后的系统，通常由软件、数据库和人员组成。软件包括操作系统、基于某种宿主语言的数据库开发工具、数据库应用系统实用程序以及数据库管理系统；数据库由数据库管理系统统一管理，数据的

插入、修改和检索均要通过数据库管理系统进行；人员包括数据库管理员、程序员和终端用户。

数据库系统一般主要由数据库、数据库管理系统、数据库开发工具、数据库应用系统和人员构成。数据库系统可以用图 1-4 来表示。

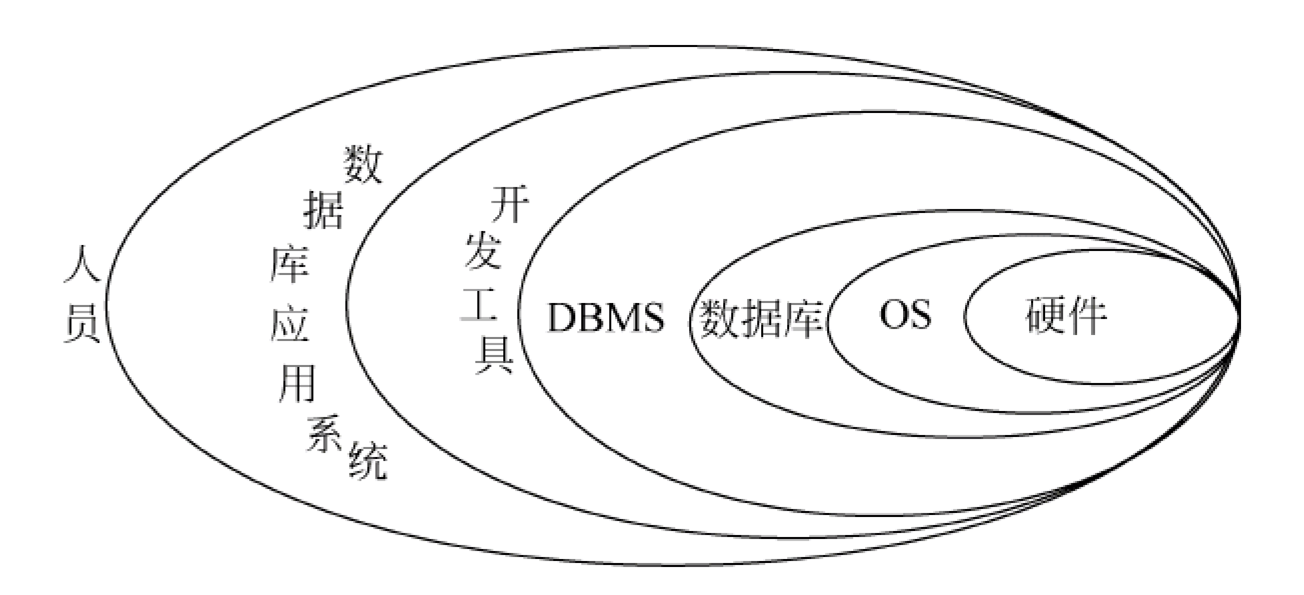


图 1-4　数据库系统的层次结构

1. 数据库

数据库(DataBase,DB)是指长期存储在计算机内,有组织的、可共享的数据集合,即在计算机系统中按一定的数据模型组织、存储和使用的相关联的数据集合。它不仅包括描述事物的数据本身,还包括相关事物之间的联系。数据库中的数据以文件的形式存储在存储介质上,是数据库系统操作的对象和结果。

2. 数据库管理系统

数据库管理系统(DBMS)是数据库系统的核心,是为数据库的建立、使用和维护而配置的软件。它建立在操作系统的基础上,是位于用户与操作系统之间的一层数据管理软件,为用户或应用程序提供访问数据库的方法,包括数据库的创建、查询、更新及各种数据控制等。数据库中数据的插入、修改和检索均要通过数据库管理系统进行,用户发出的或应用程序中的各种操作数据库中数据的命令都要通过数据库管理系统来执行。数据库管理系统还承担着数据库的维护工作,能够按照数据库管理员所规定的要求,保证数据库的安全性和完整性。

一般说来,数据库管理系统的功能主要包括以下 5 个方面。

1) 数据定义功能

DBMS 提供数据定义语言(Data Definition Language,DDL)对数据库中的对象进行定义,使用户能够定义构成数据库结构的各级模式,包括定义表结构、定义索引以及定义视图,也能够定义数据库的完整性、安全性等。这些定义存储在数据字典中,是 DBMS 运行的基本依据。

2) 数据操纵功能

DBMS 提供数据操纵语言(Data Manipulation Language,DML)操纵数据库中的数据,实现对数据库的基本操作,包括对数据库中的数据进行检索、插入、修改和删除等。

3) 数据库运行控制功能

对数据库的运行进行管理是数据库管理系统运行时的核心部分,包括对数据库进行并发控制、安全性检查、完整性约束条件的检查和执行以及数据库的内部维护等。所有访问数据库的操作都要在这些控制程序的统一管理下进行,以保证数据的安全性、完整性、一致性

以及多用户对数据库的并发使用。

4）数据库的组织、存储和管理

数据库中需要存放多种数据，如数据字典、用户数据、存取路径等，数据库管理系统负责分门别类地组织、存储和管理这些数据，确定以何种文件结构和存取方式以物理方式组织这些数据，如何实现数据之间的联系，以便提高存储空间利用率以及提高随机查找、顺序查找、增加、删除和查改等操作的时间效率。

5）建立和维护数据库

建立数据库包括数据库初始数据的输入与数据转换等。维护数据库包括数据库的备份和还原、数据库的重组织与重构造、性能的监视与分析等。

常见的数据库管理系统有 Access、SQL Server、MySQL、Oracle、DB2 等。

3. 数据库应用系统

凡使用数据库技术管理其数据的系统都称为数据库应用系统（DataBase Application System，DBAS）。数据库应用系统的应用非常广泛，它可以用于事务管理、计算机辅助设计、计算机图形分析和处理及人工智能等系统中。

4. 人员

1）终端用户

终端用户（End User）是数据库的使用者，通过应用程序与数据库进行交互。他们不需要具有数据库的专业知识，只是通过应用程序的用户接口存取数据库的数据，使用数据库来完成其业务活动，直观地显示和使用数据。

2）应用程序员

应用程序员（Application Programmer）负责分析、设计、开发、维护数据库系统中各类应用程序。数据库系统一般需要一个以上的应用程序员在开发周期内完成数据库结构设计、应用程序开发等任务，并在后期管理应用程序，保证使用周期中对应用程序在功能及性能方面的维护、修改工作。

3）数据库管理员

数据库管理员（DataBase Administrator，DBA）的职能是管理、监督、维护数据库系统的正常运行，负责全面管理和控制数据库系统。数据库管理员的主要职责包括设计与定义数据库系统，帮助最终用户使用数据库系统，监督与控制数据库系统的使用和运行，改进和重组数据库系统，优化数据库系统的性能，定义数据的安全性和完整性约束，备份与恢复数据库等。

1.1.3 数据库的数据模型表示

1. 层次模型

层次模型用树形结构来表示数据库中的数据。每个结点表示一个记录类型，结点之间的连线表示记录类型间的联系，这种联系只能是父子联系。最著名最典型的层次模型数据库系统是 IBM 公司在 1968 年开发的 IMS（Information Management System），一种适合其主机的层次模型数据库。

在这种模型中，数据被组织成由“根”开始的“树”，每个实体由根开始沿着不同的分支放在不同的层次上，如果不再向下分支，那么此分支序列中最后的结点称为“叶”。学校教学机

构的层次模型如图 1-5 所示。

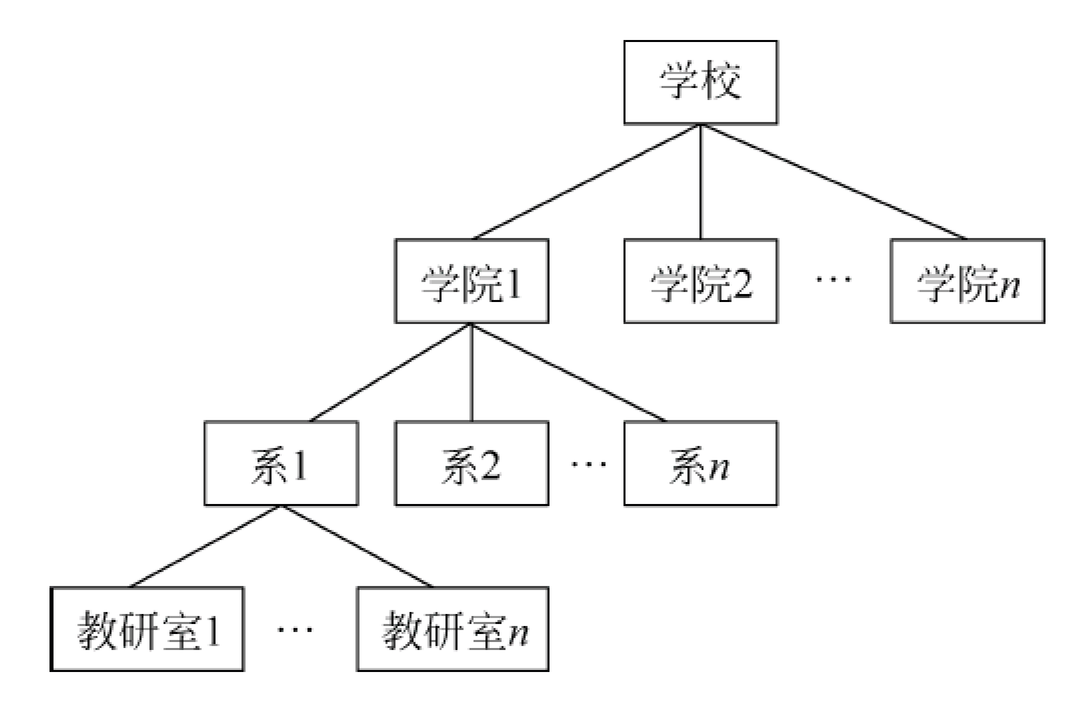


图 1-5 学校教学机构的层次模型

层次模型的优点是模型本身比较简单，只需很少几条命令就能操纵数据库；对于记录间联系是固定的，且预先定义好的应用系统，采用层次模型最易实现。但其缺点也很多，如插入和删除操作的限制比较多；查询子女结点必须通过双亲结点等。

2. 网状模型

1964 年通用电气公司（General Electric Co.）的 Charles Bachman 成功地开发出世界上第一个网状模型数据库管理系统，即第一个数据库管理系统——集成数据存储（Integrated Data Store，IDS），奠定了网状模型数据库的基础，并在当时得到了广泛的发行和应用。

网状数据模型是一种比层次模型更具普遍性的结构，从图论的观点看，它是一个不加任何条件的无向图。一般来说，层次模型是网状模型的特殊形式，网状模型是层次模型的一般形式。学生选课系统的网状模型如图 1-6 所示。

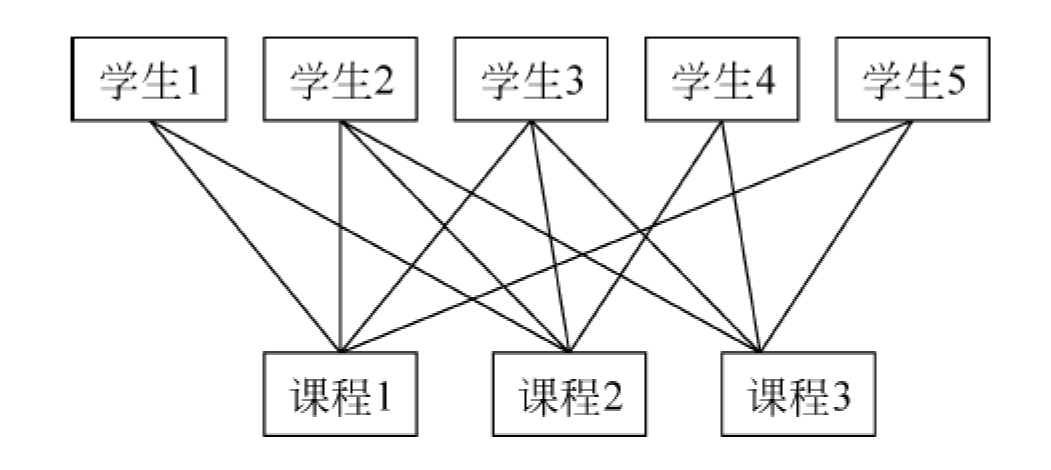


图 1-6 学生选课系统的网状模型

网状模型与层次模型相比，提供了更大的灵活性，能更直接地描述现实世界，性能和效率也比较好。网状模型的缺点是结构比较复杂，不易于掌握；DDL 和 DML 复杂，不易于使用；记录之间联系变动后涉及链接指针的调整，扩充和维护都比较复杂。

3. 关系模型

对于两种非关系模型，对数据的操作是过程化的，应用程序在访问数据时要指定存取路径。而后来出现的关系模型较好地解决了这些问题。1970 年，IBM 的研究员 E. F. Codd 博士在刊物 *Communication of the ACM* 上发表了一篇名为 *A Relational Model of Data for Large Shared Data Banks* 的论文，提出了关系模型的概念，奠定了关系模型的理论基础。

用二维表格结构表示数据库中数据的模型称为关系模型。关系模型在用户看来就是二维表，其概念单一，容易被初学者接受。在关系模型中，操作的对象和操作结果都是二维表。

下面以表 1-1 所示的“职工信息表”为例介绍关系模型中的几个重要概念。

表 1-1 职工信息表

职工号	姓名	性别	年龄	工资
200331	张　玉	男	50	6000
200332	黎　明	男	40	4900
200334	王　洪	男	33	4600
200346	赵小溪	女	42	5000

(1) 关系：一个关系就是一张二维表，每个关系都有一个关系名，如表 1-1 就是一个职工信息关系表。在计算机中，一个关系可以存储为一个文件。

(2) 元组：二维表中的行称为元组，每一行是一个元组。元组对应存储文件中的一个记录，职工信息表中包括 4 个元组。

(3) 属性：二维表的列称为属性，每一列有一个属性名，属性值是属性的具体取值。属性对应存储文件中的一个字段。职工信息表中包括 5 个属性，属性名分别是职工号、姓名、性别、年龄和工资，属性的具体取值就形成表中的一个个元组。

(4) 域：域是属性的取值范围。例如，职工信息表中性别的取值范围只能是男和女，即性别的域为(男，女)。

(5) 关系模式：对关系的信息结构及语义限制的描述称为关系模式，用关系名和所包含的属性名的集合表示。例如，职工信息表的关系模式是：职工(职工号，姓名，性别，年龄，工资)，属性间用逗号间隔。

(6) 关键字或码：在关系的属性中，能够用来唯一标识元组的属性(或属性组合)称为关键字或码。

(7) 候选关键字或候选码：如果在一个关系中，存在多个属性(或属性组合)都能用来唯一标识该关系中的元组，这些属性(或属性组合)都称为该关系的候选关键字或候选码，候选码可以有多个。例如，在职工信息表中，如果没有重名的元组，则职工号和姓名都是职工信息表的候选码。

(8) 主键或主码：在一个关系的若干候选关键字中，被指定作为关键字的候选关键字称为该关系的主键或主码(Primary Key)。一般地，习惯选择号码作为一个关系的主码，比如在职工信息表中，一般会选择职工号作为该关系的主码；当然，在姓名也是候选码的情况下，也可以选择姓名作为该关系的主码。但是，一个关系的主码，在同一时刻只能有一个。

(9) 主属性和非主属性：在一个关系中，包含在任何候选关键字中的属性都称为主属性；不包含在任何候选关键字中的属性都称为非主属性。例如，职工信息表中的职工号和姓名是主属性，而性别、年龄和工资是非主属性。

(10) 外键或外码：一个关系的某个属性(或属性组合)不是该关系的主码或只是主码的一部分，却是另一个关系的主码，则称这样的属性为该关系的外键或外码(Foreign Key)。外码是表与表联系的纽带。

1.2 SQL Server 2008 基础

SQL Server 是美国 Microsoft 公司推出的一种关系数据库系统，是一个可扩展的、高性能的、为分布式客户机/服务器计算所设计的数据库管理系统，实现了与 Windows NT 的有

机结合，提供了基于事务的企业级信息管理系统方案。

跟以往版本的产品相比，Microsoft SQL Server 2008 被定位为可用性和大数据领域的领头羊；对数据的转换和挖掘提供强大的交互操作能力，并协助做出正确的决策；具有开放性、可伸缩性、安全性、可扩展性以及前所未有的高性能等优势。

SQL Server 提供了图形和命令行工具，用户可以使用不同的方法访问数据库，但这些工具的核心是 T-SQL 语言。

本章主要介绍 SQL Server 的发展史，SQL Server 2008 的新增功能、系统架构和协议，SQL Server 2008 安装的软、硬件需求及安装过程，SQL Server 2008 的主要组件、基本使用方法，以及 T-SQL 语言基础知识。

1.2.1 SQL Server 发展史

我们通常把 Microsoft SQL Server 简称为 SQL Server，但事实上，最早的 SQL Server 系统并不是 Microsoft 公司开发出来的，而是由赛贝斯公司推出的。

1987 年，赛贝斯公司发布了 Sybase SQL Server 系统。

1988 年，Microsoft 公司、Aston-Tate 公司参加到了赛贝斯公司的 SQL Server 系统开发中。

1989 年，推出了 SQL Server 1.0 for OS/2 系统。

1990 年，Aston-Tate 公司退出了联合开发团队，Microsoft 公司则希望将 SQL Server 移植到自己刚刚推出的新技术产品即 Windows NT 系统中。

1992 年，Microsoft 公司与赛贝斯公司签署了联合开发用于 Windows NT 环境的 SQL Server 系统的协议。

1993 年，Microsoft 公司与赛贝斯公司在 SQL Server 系统方面的联合开发正式结束。

1995 年，Microsoft 公司成功地发布了 Microsoft SQL Server 6.0 系统。

1996 年，Microsoft 公司又发布了 Microsoft SQL Server 6.5 系统。

1998 年，Microsoft 公司又成功地推出了 Microsoft SQL Server 7.0 系统。

2000 年，Microsoft 公司迅速发布了与传统 SQL Server 有重大不同的 Microsoft SQL Server 2000 系统。

2005 年 12 月，Microsoft 公司发布了 Microsoft SQL Server 2005 系统，其高效的数据处理、强大的功能、简易而统一的界面操作，受到众多软件公司和企业的青睐。

2008 年 8 月，Microsoft 公司发布了 Microsoft SQL Server 2008 系统，其代码名称是 Katmai，在安全性、可用性、易管理性、可扩展性、商业智能等方面有了更多的改进和提高，对企业的数据存储和应用需求提供了更强大的支持和便利。

2012 年 3 月，Microsoft 公司发布了 Microsoft SQL Server 2012 系统，除保留了 SQL Server 2008 的风格外，在安全性、高可用性、多维数据分析、报表分析以及大数据的支持等方面有了较大的提高和突破。

Microsoft SQL Server 的最新产品为 SQL Server 2016，和以前的版本相比改进较大，是 Microsoft 数据平台历史上最大的一次跨越性发展。

目前，普通高校一般都在使用 SQL Server 2008 作为教学版本，所以本书以 SQL Server 2008 数据平台为基础介绍数据库的操作、管理和安全问题。

1.2.2 SQL Server 2008 新增功能

Microsoft 数据平台提供了一个解决方案来存储和管理许多数据类型，包括 XML、E-mail、时间/日历、文件、文档、地理信息等；同时提供一个丰富的服务集合来与数据进行交互，实现搜索、查询、数据分析、报表、数据整合和同步功能。SQL Server 2008 给出了如图 1-7 所示的平台。

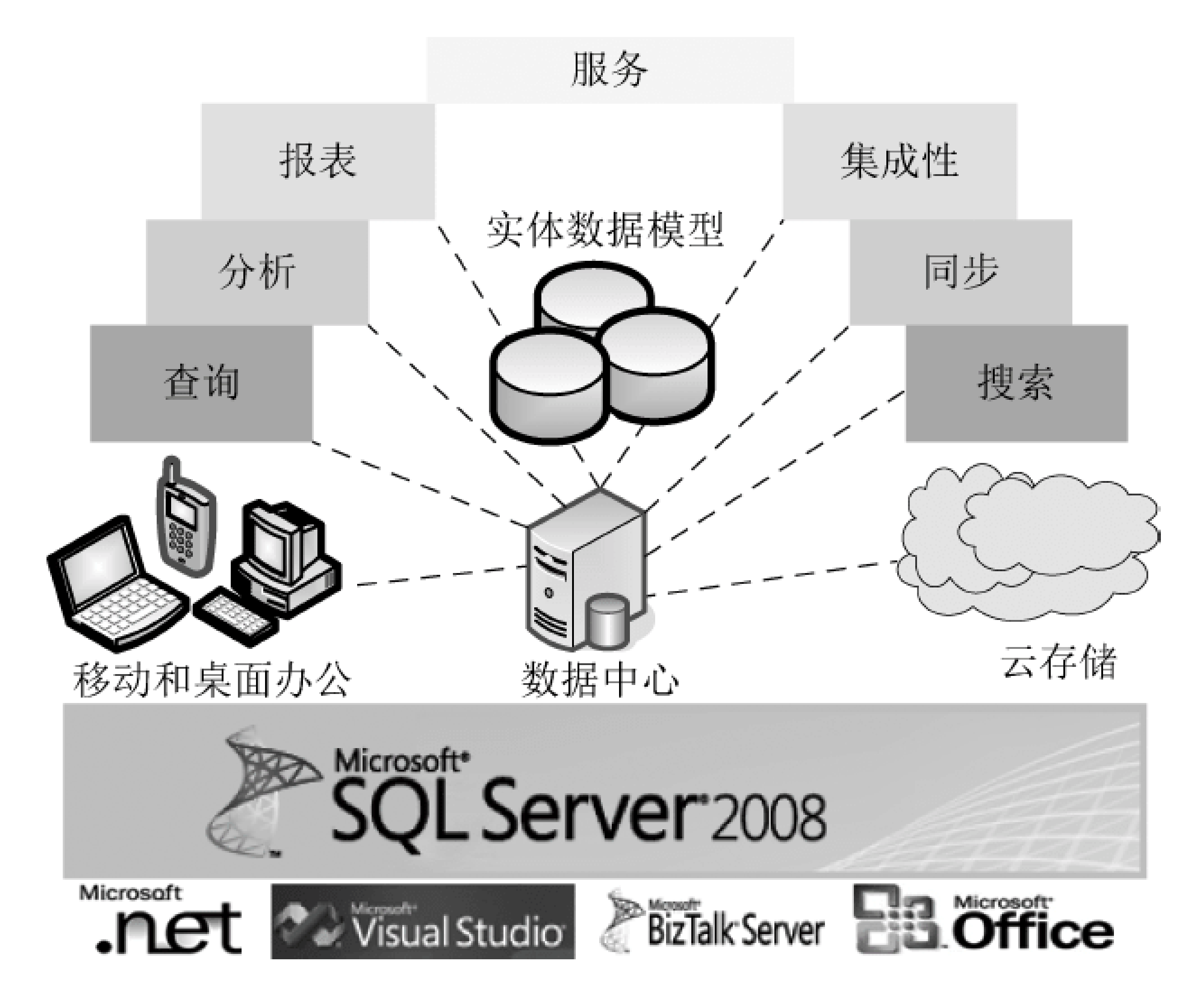


图 1-7 Microsoft 数据平台

SQL Server 2008 出现在 Microsoft 数据平台上，使得公司可以运行它们最关键的应用程序，同时降低了用户管理数据基础设施和发送观察信息的成本。该平台具有以下特点。

1. 可信任的

在今天以数据驱动的世界中，公司需要持续访问其数据。SQL Server 2008 为关键任务应用程序提供了强大的安全性、可靠性和可扩展性。

1) 安全性

在过去的 SQL Server 2005 的基础之上，SQL Server 2008 做了以下方面的增强来扩展它的安全性：

(1) 简单的数据加密。SQL Server 2008 可以对整个数据库、数据文件和日志文件进行加密，而不需要改动应用程序。加密使公司可以满足遵守规范和关注数据隐私的要求。简单的数据加密的好处包括使用任何范围或模糊查询搜索加密的数据、加强数据安全性以防止未授权的用户访问以及数据加密。这些都可以在不改变已有的应用程序的情况下进行。

(2) 外键管理。SQL Server 2008 为加密和密钥管理提供了一个全面的解决方案。为了满足不断发展的对数据中心的信息的更强安全性的需求，公司投资给供应商来管理公司内的安全密钥。SQL Server 2008 通过支持第三方密钥管理和硬件安全模块产品为这个需求提供了很好的支持。

(3) 增强了审查。SQL Server 2008 使用户可以审查数据的操作，从而提高了遵从性和安全性。审查不只针对数据修改的所有信息，还包括关于什么时候对数据进行读取的信息。

SQL Server 2008 具有像服务器中加强的审查的配置和管理这样的功能，这使得公司可以满足各种规范需求。SQL Server 2008 还可以定义每一个数据库的审查规范，所以审查配置可以针对每一个数据库单独制定。为指定对象作审查配置使审查的执行性能更好，配置的灵活性也更高。

2）可靠性

(1) 改进了数据库镜像。SQL Server 2008 基于 SQL Server 2005，并提供了更可靠的加强了数据库镜像的平台，包括页面自动修复、压缩输出的日志流、新增加的执行计数器，以及动态管理视图和对现有视图的扩展。

(2) 热添加 CPU。为了在线添加内存资源而扩展 SQL Server 中已有的支持，热添加 CPU 使数据库可以按需扩展。事实上，CPU 资源可以添加到 SQL Server 2008 所在的硬件平台上而不需要停止应用程序。

3）可扩展性

公司在面对不断增长的压力，要提供可预计的响应和对随着用户数目的增长而不断增长的数据量进行管理。SQL Server 2008 提供了一个广泛的功能集合，使数据平台上的所有工作负载的执行都是可扩展的和可预测的。

(1) 性能数据的采集。性能调整和故障排除对于管理员来说是耗费时间的工作。为了给管理员提供全面的执行洞察力，SQL Server 2008 推出了范围更大的数据采集、一个用于存储性能数据的新的集中的数据库，以及新的报表和监控工具。

(2) 扩展事件。SQL Server 扩展事件是一个用于服务器系统的一般的事件处理系统。扩展事件基础设施是一个轻量级的机制，它支持对服务器运行过程中产生的事件的捕获、过滤和响应。这个对事件进行响应的能力使用户可以通过增加前后文关联数据，例如 T-SQL 对所有事件调用堆栈或查询计划句柄，以此来快速诊断运行时问题。事件捕获可以按几种不同的类型输出，包括 Windows 事件跟踪(Event Tracing for Windows，ETW)。当扩展事件输出到 ETW 时，操作系统和应用程序就可以关联了，这使得用户可以进行更全面的系统跟踪。

(3) 备份及数据压缩。保持在线进行基于磁盘的备份是很昂贵而且很耗时的。有了 SQL Server 2008 备份压缩，需要的磁盘 I/O 减少了，在线备份所需要的存储空间也减少了，而且备份的速度明显加快了。改进的数据压缩使数据可以更有效地存储，并且降低了数据的存储要求。数据压缩还为大型的限制输入/输出的工作负载(例如数据仓库)提供了显著的性能改进。

另外，随着资源监控器的推出，SQL Server 2008 使公司可以提供持续的和可预测的响应给终端用户；还提供了一个新的制订查询计划的功能，从而提供了更好的查询执行稳定性和可预测性，使公司可以在硬件服务器更换、服务器升级和产品部署中提供稳定的查询计划。

2. 高效的

SQL Server 2008 降低了开发和管理数据基础设施的时间和成本，使得开发人员可以开发强大的下一代数据库应用程序。

1）基于策略的管理

作为 Microsoft 正在努力降低公司的总成本所做的工作的一部分，SQL Server 2008 推

出了陈述式管理架构(DMF),它是一个用于 SQL Server 数据库引擎的新的基于策略的管理框架。

DMF 是一个基于策略的用于管理一个或多个 SQL Server 2008 实例的系统。要使用 DMF,SQL Server 策略管理员使用 SQL Server 管理套件创建策略,这些策略管理服务器上的实体,例如 SQL Server 的实例、数据库和其他 SQL Server 对象。DMF 由 3 个组件组成:策略管理、创建策略的策略管理员和显式管理。管理员选择一个或多个要管理的对象,并显式检查这些对象是否遵守指定的策略,或显式地使这些对象遵守某个策略。

2) 改进了安装

SQL Server 2008 对 SQL Server 的服务生命周期提供了显著的改进,它重新设计了安装、建立和配置架构。这些改进将计算机上的各个安装与 SQL Server 软件的配置分离开来,这使得公司和软件合作伙伴可以提供推荐的安装配置。

3) 加速开发过程

SQL Server 提供了集成的开发环境和更高级的数据提取功能,使开发人员可以创建下一代数据库应用程序,同时简化了对数据的访问。

(1) ADO. NET 实体框架。数据库开发的一个趋势是定义高级的业务对象或实体,然后可以将它们匹配到数据库中的表和字段,开发人员使用高级实体例如"客户"或"订单"来显示背后的数据。ADO. NET 实体框架使开发人员可以以这样的实体来设计关系数据。在这一提取级别的设计是非常高效的,开发人员可以充分利用。

(2) 实体关系建模。Microsoft 的语言级集成查询能力(LINQ)使开发人员可以通过使用管理程序语言,例如 C# 或 Visual Basic. NET,而不是 SQL 语句,来对数据进行查询。LINQ 使得用. NET 框架语言编写的无缝和强大的面向集合的查询可以运行于 ADO. NET (LINQ 到 SQL)、ADO. NET 数据集(LINQ 到数据集)、ADO. NET 实体框架(LINQ 到实体),和到实体数据服务给予匹配的供应商。SQL Server 2008 提供了一个新的 LINQ 到 SQL 供应商,使得开发人员可以直接将 LINQ 用于 SQL Server 2008 的表和字段。

ADO. NET 的对象服务层使得可以进行具体化检索、改变跟踪和实现可持续性地为公共语言运行时(CLR)的数据。开发人员使用 ADO. NET 实体框架可以通过使用由 ADO . NET 管理的 CLR 对象对数据库进行编程。SQL Server 2008 提供了提高性能和简化开发过程的更有效的和最佳的支持。

另外,SQL Server 2008 继续加强了 Service Broker 的能力;对 T-SQL 进行了改进,通过几个关键的改进增强了 T-SQL 编程人员的开发体验。

4) 偶尔连接系统

有了移动设备和移动工作人员,偶尔连接成为一种工作方式。SQL Server 2008 推出了一个统一的同步平台,使得在应用程序、数据存储和数据类型之间达到一致性同步。在与 Visual Studio 的合作下,SQL Server 2008 使得可以通过 ADO. NET 中提供的新的同步服务和 Visual Studio 中的脱机设计器快速创建偶尔连接系统。SQL Server 2008 提供了支持,使得可以改变跟踪和使客户可以以最小的执行消耗进行功能强大的操作,以此来开发基于缓存的、基于同步的和基于通知的应用程序。

5) 不只是关系数据

应用程序正在结合使用越来越多的数据类型,而不仅仅是过去数据库所支持的那些。

SQL Server 2008 基于过去对非关系数据的强大支持，提供了新的数据类型，使得开发人员和管理员可以有效地存储和管理非结构化数据，例如文档和图片；还增加了对管理高级地理数据的支持。

3. 智能的

SQL Server 2008 在整个企业范围内实现了全面的商务智能(BI)，可进行任意大小、任意复杂度的报表和数据分析，实现强大的界面交互并与 Microsoft Office System 高度集成。

1）集成任何数据

Microsoft 继续投资于商务智能和数据仓库解决方案，以便从其数据中获取商业价值。SQL Server 2008 提供了一个全面的和可扩展的数据仓库平台，它可以用一个单独的分析存储进行强大的分析，以满足成千上万的用户在几兆字节的数据中的需求。

2）发送相应的报表

SQL Server 2008 提供了一个可扩展的商务智能基础设施，使得 IT 人员可以在整个公司内使用商务智能来管理报表以及进行任何规模和复杂度的分析。SQL Server 2008 使得公司可以有效地以用户想要的格式和他们的地址发送相应的、个人的报表给成千上万的用户。通过提供交互发送用户需要的企业报表，获得报表服务的用户数目大大增加了。这使得用户可以获得他们各自领域的相关信息，进而做出更好、更快、更符合要求的决策。

3）使用户获得全面的洞察力

及时访问准确信息，使用户快速对问题甚至是非常复杂的问题做出反应，这是在线分析处理的前提(Online Analytical Processing，OLAP)。SQL Server 2008 基于 SQL Server 2005 强大的 OLAP 能力，为所有用户提供了更快的查询速度。这个性能的提升使得公司可以执行具有许多维度和聚合的非常复杂的分析。这个执行速度与 Microsoft Office 的深度集成相结合，使 SQL Server 2008 可以让所有用户获得全面的洞察力。

1.2.3 SQL Server 2008 的新特性

SQL Server 2008 不仅对原有性能进行了改进，还添加了许多新特性，例如新添了数据集成功能，改进了分析服务、报表服务以及 Office 集成等。

1. SQL Server 集成服务

SQL Server 集成服务(SQL Server Integration Services，SSIS)是一个嵌入式应用程序，用于开发和执行 ETL(Extract-Transform-Load，解压缩、转换和加载)包。SSIS 代替了 SQL Server 2000 的 DTS(Data Transformation Services，数据转换服务)，其集成服务功能既包含了实现简单的导入导出包所必需的 Wizard 导向插件、工具以及任务，也有非常复杂的数据清理功能。

2. 分析服务

SQL Server 分析服务(SQL Server Analysis Services，SSAS)也得到了很大的改进和增强。其中 IB 堆叠做出了改进，性能得到很大提高，而硬件商品能够为 Scale out 管理工具所使用，Block Computation 也增强了立体分析的性能。

3. 报表服务

SQL Server 报表服务(SQL Server Reporting Services，SSRS)的处理能力和性能得到改进，使得大型报表不再耗费所有可用内存。另外，在报表的设计和完成之间有了更好的一

致性。SQL SSRS 2008 还包含了跨越表格和矩阵的 Tablix。Application Embedding 允许用户单击报表中的 URL 链接调用应用程序。

4. Microsoft Office 2007

SQL Server 2008 能够与 Microsoft Office 2007 完美结合。例如,SSRS 能够直接把报表导出成为 Word 文档。而且 Report Authoring 工具、Word 和 Excel 都可以作为 SSRS 报表的模板。Excel SSAS 新添了一个数据挖掘插件,提高了其性能。

1.2.4 SQL Server 2008 的版本

SQL Server 2008 版本很多,根据需求,可以选择的 SQL Server 2008 版本也各不相同,而根据应用程序的需要,安装要求亦会有所不同。不同版本的 SQL Server 能够满足单位和个人独特的性能、运行时以及价格要求。安装哪些 SQL Server 组件还取决于用户具体需要。

SQL Server 2008 分为 SQL Server 2008 企业版、标准版、工作组版、Web 版、开发者版、Express 版、Compact 3.5 版,其功能和作用也各不相同,其中 SQL Server 2008 Express 版是免费版本。

1. SQL Server 2008 企业版

SQL Server 2008 企业版是一个全面的数据管理和商务智能平台,为关键业务应用提供了企业级的可扩展性、数据仓库、安全、高级分析和报表支持。这一版本将为用户提供更加坚固的服务器和执行大规模在线事务处理的能力。

2. SQL Server 2008 标准版

SQL Server 2008 标准版是一个完整的数据管理和商务智能平台,为部门级应用提供了最佳的易用性和可管理特性。

3. SQL Server 2008 工作组版

SQL Server 2008 工作组版是一个值得信赖的数据管理和报表平台,用于实现安全的发布、远程同步和对运行分支应用的管理能力。这一版本拥有核心的数据库特性,可以很容易地升级到标准版或企业版。

4. SQL Server 2008 Web 版

SQL Server 2008 Web 版是针对运行于 Windows 服务器中要求高可用、面向 Internet Web 服务的环境而设计的。这一版本为实现低成本、大规模、高可用性的 Web 应用或客户托管解决方案提供了必要的支持工具。

5. SQL Server 2008 开发者版

SQL Server 2008 开发者版允许开发人员构建和测试基于 SQL Server 的任意类型应用。这一版本拥有所有企业版的特性,但只限于在开发、测试和演示中使用。基于这一版本开发的应用和数据库可以很容易地升级到企业版。

6. SQL Server 2008 Express 版

SQL Server 2008 Express 版是 SQL Server 的一个免费版本,它拥有核心的数据库功能,其中包括了 SQL Server 2008 中最新的数据类型,但它是 SQL Server 的一个微型版本。这一版本是为了学习、创建桌面应用和小型服务器应用而发布的,也可供 ISV 再发行使用。

7. SQL Server Compact 3.5 版

SQL Server Compact 是一个针对开发人员而设计的免费嵌入式数据库，这一版本的意图是构建独立、仅有少量连接需求的移动设备、桌面和 Web 客户端应用。SQL Server Compact 可以运行于所有的 Windows 平台之上，包括 Windows XP 和 Windows Vista 操作系统，以及 Pocket PC 和 SmartPhone 设备上。

大型的企业客户大多希望以一种简洁的方式获得一个完整的、集成的数据平台，他们希望使用一个能够满足各方面需求的数据库产品，所以 SQL Server 2008 企业版是这部分客户的理想选择。

对于中小型的企业客户，使用 SQL Server 2008 标准版完全能够满足其需求。而对于小型机构或个人，SQL Server 2008 工作组版快捷易用，是入门级的数据库产品；当然，如果考虑到成本的问题，Express 版也是很好的选择。

对于数据库开发人员而言，可以根据不同情况来选择开发者版或 Compact 3.5 版。SQL Server 2008 Web 版的性能要低于企业版和标准版，但对于 Web 宿主和网站的开发而言，它是一个低成本、高可用性的选择。

1.2.5 SQL Server 2008 的环境需求与安装

1. 环境需求

环境需求是指系统安装时对硬件、操作系统、网络等环境的要求，这些要求也是 Microsoft SQL Server 系统运行所必需的条件。

若根据现今的最低硬件规格标准来判断，哪怕是最低成本的方案，多数情况下对于大部分的 SQL Server 版本，都可以满足运行的要求。但为保险起见仍应了解一下最低硬件配置是什么，并以此检查所拥有的计算机，确认其具备满足需求的硬件资源。

1）CPU

对于运行 SQL Server 的 CPU，建议的最低要求是 32 位版本对应 1GHz 的处理器，64 位版本对应 1.6GHz 的处理器，或兼容的处理器，或具有类似处理能力的处理器，但推荐使用 2GHz 的处理器。然而，像这里列出的大多数最低配置要求一样，Microsoft 事实上推荐的是更快的处理器。处理器越快，SQL Server 运行得就越好，由此而产生的瓶颈也越少。现在的很多计算机使用的都是 2GHz 以上的处理器。这将缩减开发所花费的时间。

另外，与提升 SQL Server 的运行速度相关的硬件并非只有处理器，SQL Server 的速度在很大程度上也受当前计算机中内存空间的影响。所以内存最低要求是 1GB，一般最好是 4GB 以上。

2）硬盘空间

SQL Server 需要比较大的硬盘空间。这不足为奇，如今主要的应用程序都是需要大量的硬盘空间的。不考虑要添加的数据文件，SQL Server 自身将占用 1GB 以上的硬盘空间。当然，本章后面要用到的安装选项将决定总共所需的硬盘空间。通过选择不安装某个可选部件，可以减少对硬盘空间的需求，例如，选择不安装联机丛书。不过，如今硬盘空间是相对廉价的，因此，最好是购买容量远远超出当前所需容量的硬盘，而不要采用恰好满足眼下空间大小要求的硬盘，以免将来不得不另行购买硬盘以满足增长的要求，这样将带来移动资料、整理原先硬盘上的空间等问题。

此外，还需要在硬盘上留有备用的空间，以满足 SQL Server 和数据库的扩展。另外，还需要为开发过程中要用到的临时文件准备硬盘空间。

3）操作系统需求

SQL Server 2008 可以运行在 Windows Vista Home Basic 及更高版本上，也可以在 Windows XP 上运行。从服务器端来看，它可以运行在 Windows Server 2003 SP2 及 Windows Server 2008 上。它也可以运行在 Windows XP Professional 的 64 位操作系统上，以及 Windows Server 2003 和 Windows Server 2008 的 64 位版本上。因此，可以运行 SQL Server 的操作系统是很多的。

4）其他软件需求

需要 Microsoft Windows Installer 3.1 或更高版本以及 Microsoft 数据访问组件(MDAC) 2.8 SP1 或更高版本。需要 Microsoft Internet Explorer 6.0 SP1 或更高版本。

2. 安装过程

安装是使用任何软件系统之前必须做的事情，是使用软件系统的开始。正确地安装和配置系统，是确保软件系统安全、健壮运行的基础工作。

若使用光盘进行安装，则插入 SQL Server 2008 的安装光盘，然后双击根目录中的 setup.exe 程序。如果不使用光盘进行安装，则双击下载的可执行安装程序即可。以下是在 Windows 7 平台上安装 SQL Server 2008 Express 版的主要步骤。

注意：Microsoft SQL Server 2008 与 Windows 7 操作系统存在一定的兼容性问题，在完成安装之后需要为 Microsoft SQL Server 2008 安装 SP1 补丁。其他版本的操作系统不存在此问题。

(1) 当安装程序启动后，首先检测是否有.NET Framework 3.5 环境。如果没有，会弹出安装此环境的对话框，此时可以根据提示安装.NET Framework 3.5。

(2) 在 Windows 7 操作系统中，启动 Microsoft SQL Server 2008 安装程序后，系统兼容性助手将提示软件存在兼容性问题，在安装完成之后必须安装 SP1 补丁才能运行，如图 1-8 所示。这里单击“运行程序”按钮开始 SQL Server 2008 的安装。



图 1-8　兼容性问题提示

(3) 进入 SQL Server 安装中心后跳过“计划”内容，直接选择界面左侧列表中的“安装”选项，进入安装列表选择。如图 1-9 所示，进入“SQL Server 安装中心”窗口后，右侧的列表显示了不同的安装选项。

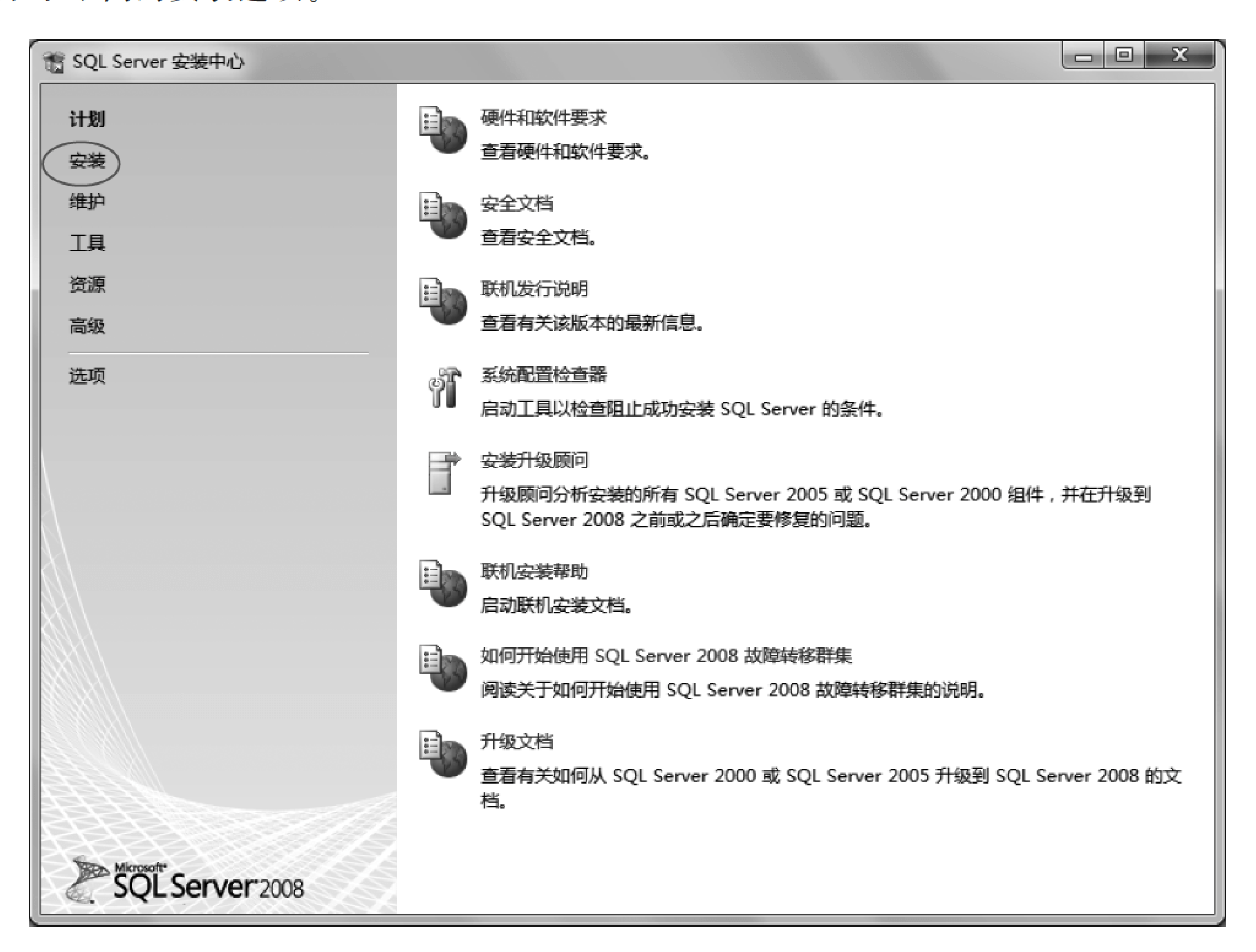


图 1-9 “SQL Server 安装中心”窗口

(4) 选择全新安装之后，系统程序兼容助手再次提示兼容性问题，如图 1-10 所示。单击“运行程序”按钮继续安装。



图 1-10 兼容性问题提示

(5) 单击“安装”按钮，进入“安装程序支持规则”安装界面，安装程序将自动检测安装环境基本支持情况，需要保证通过所有条件后才能进行下面的安装，当完成所有检测后，如图 1-11 所示，单击“确定”按钮进行下面的安装。



图 1-11 “安装程序支持规则”窗口

(6) 首先安装“安装程序支持文件”，如图 1-12 所示，这些文件是必需的。



图 1-12 “安装程序支持文件”窗口

(7) 再次进入"安装程序支持规则"窗口进行下一步检测，当所有检测都通过之后才能继续下面的安装，如图 1-13 所示；如果出现错误，需要更正所有失败后才能继续安装。



图 1-13 "安装程序支持规则"窗口

(8) 单击"下一步"按钮，进入"安装类型"窗口，选中"执行 SQL Server 2008 的全新安装"单选按钮，如图 1-14 所示。单击"下一步"按钮。

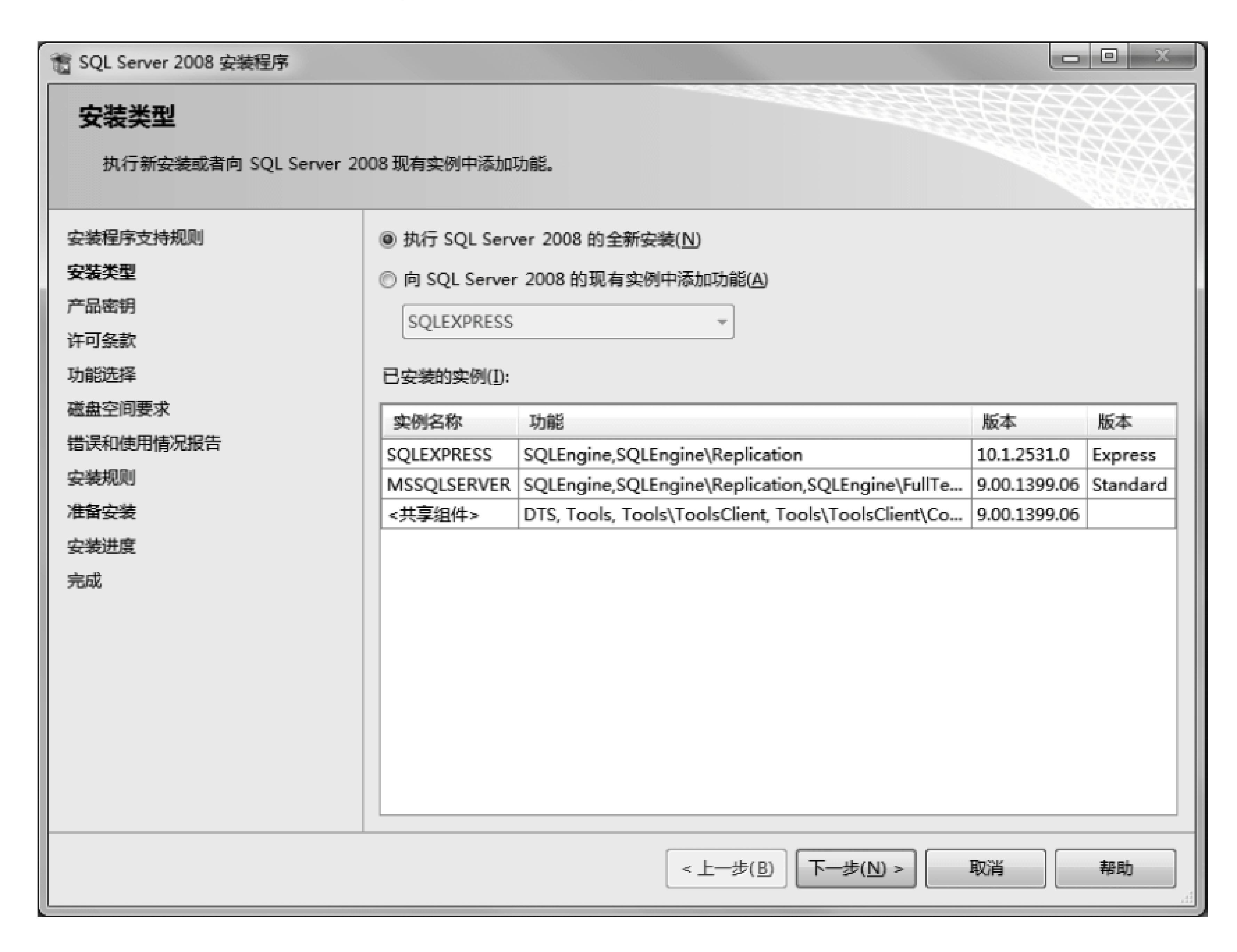


图 1-14 "安装类型"窗口

(9) 在下面的窗口中需要进行 SQL Server 2008 版本选择或填写密钥，Enterprise Evaluation 的安装密钥可以通过 Microsoft 官方购买。目前安装的是具有高级服务的 Express 版本，在“指定可用版本”处可以选择，如图 1-15 所示。

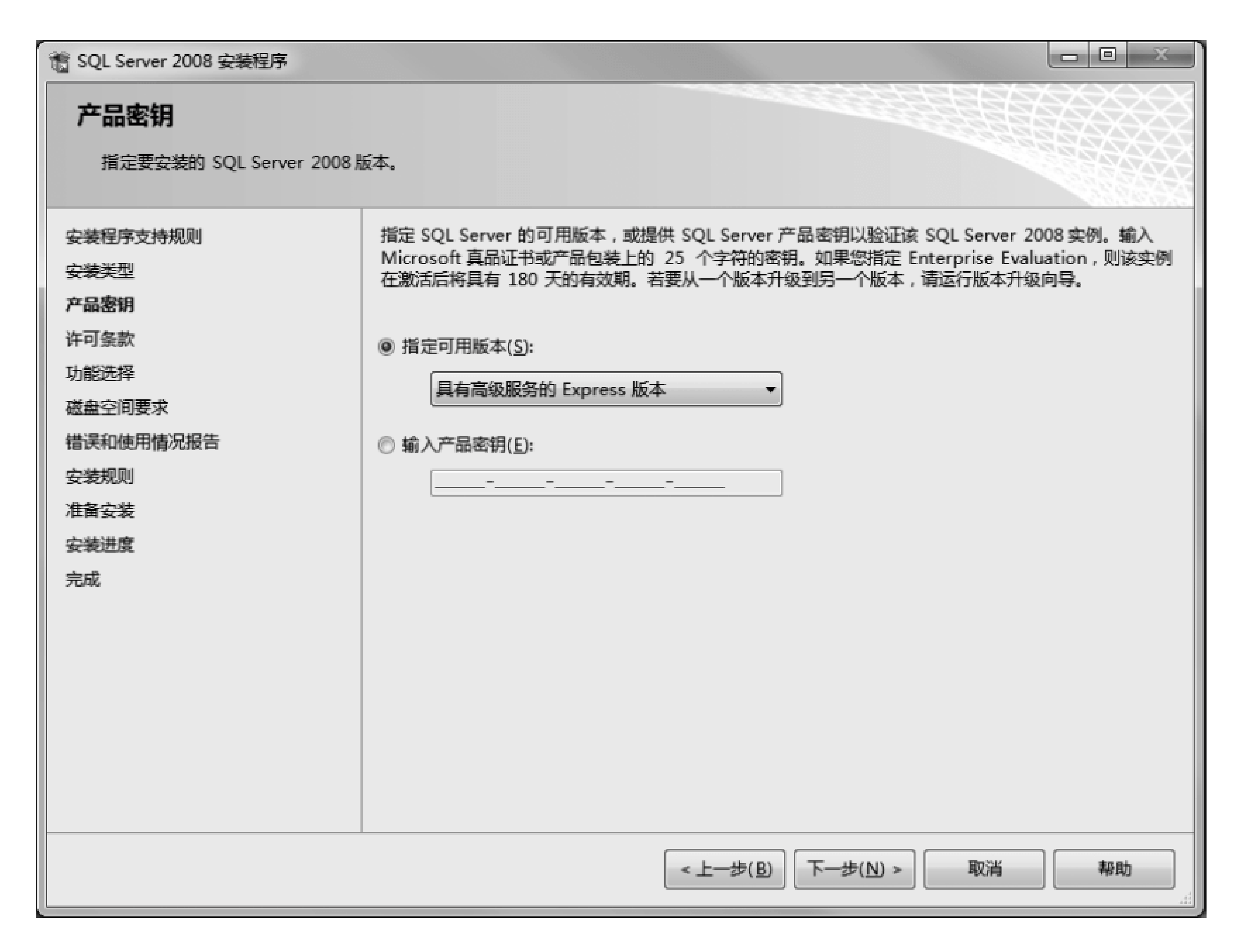


图 1-15 “产品密钥”窗口

(10) 单击“下一步”按钮，在“许可条款”窗口中，选择接受 Microsoft 软件许可条款，然后才能继续安装 SQL Server 2008，如图 1-16 所示。单击“下一步”按钮。



图 1-16 “许可条款”窗口

(11) 根据需要选择要安装的功能,可以单击"全选"按钮,如图 1-17 所示。单击"下一步"按钮。



图 1-17 "功能选择"窗口

(12) 进入"实例配置"窗口,可以选择"默认实例",也可以选择"命名实例",再选择实例要安装的路径,如图 1-18 和图 1-19 所示。

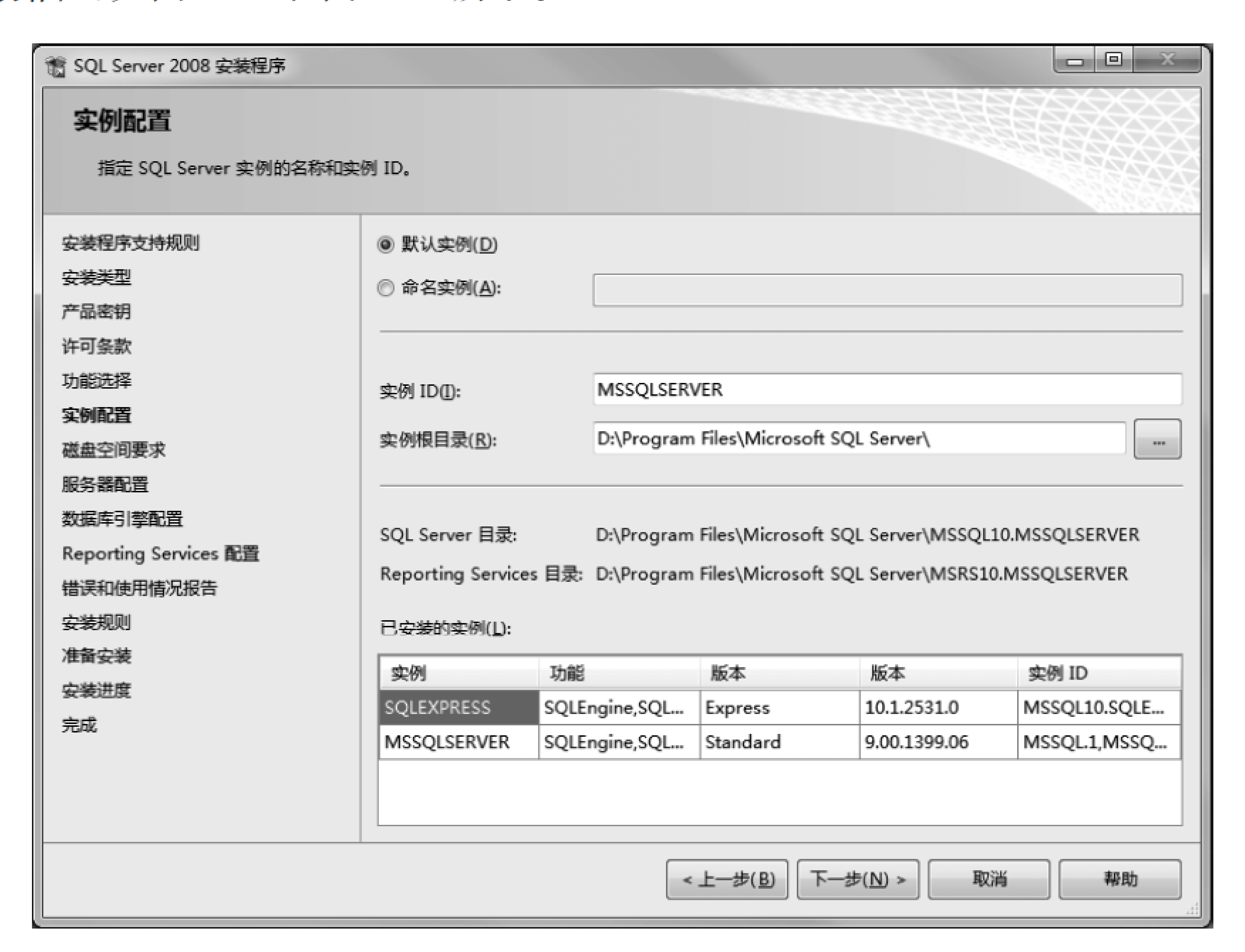


图 1-18 "实例配置"窗口

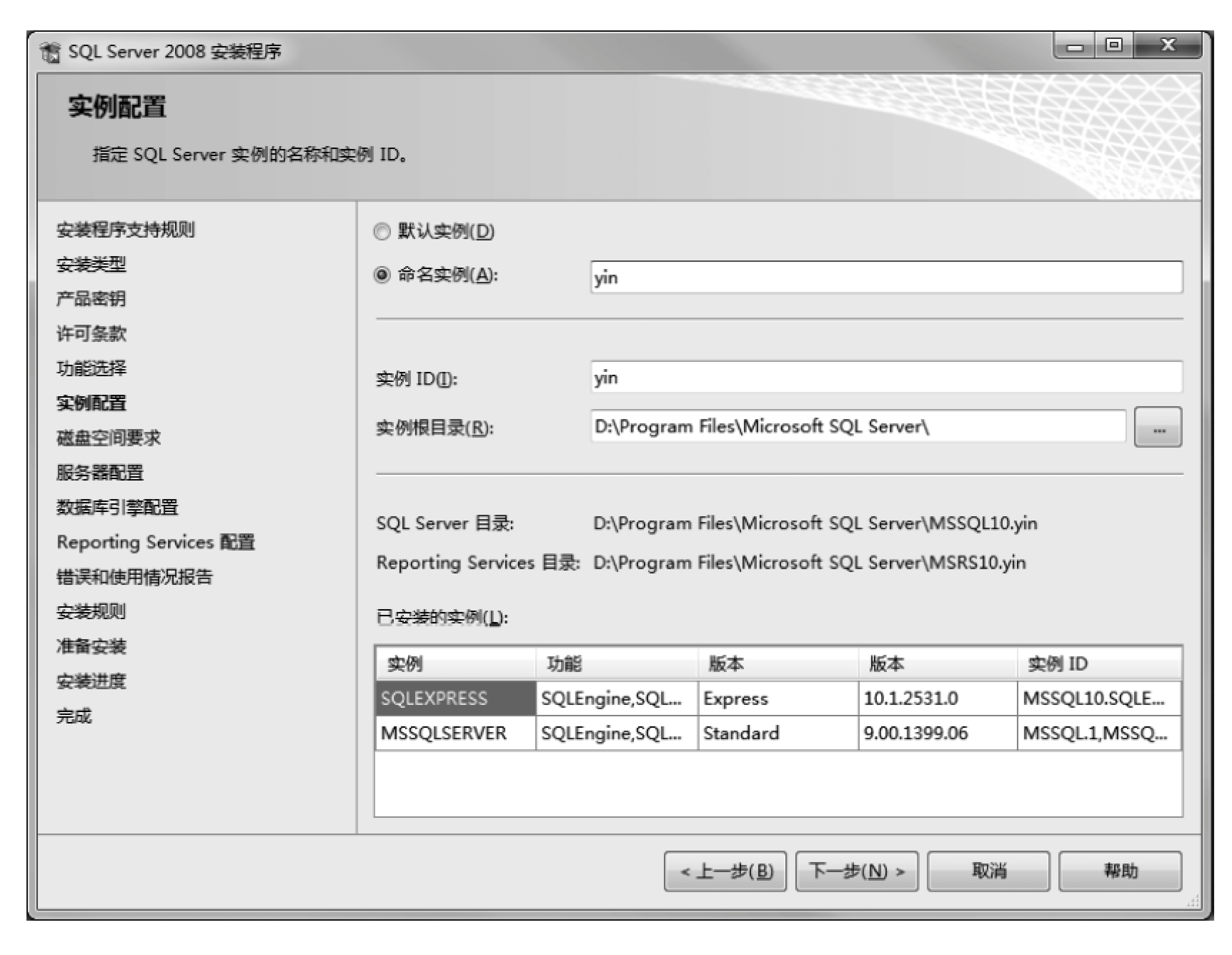


图 1-19 “实例配置”窗口

(13) 单击“下一步”按钮,进入“磁盘空间要求”窗口,了解磁盘可用情况和本软件的需求,如图 1-20 所示。



图 1-20 “磁盘空间要求”窗口

(14) 单击“下一步”按钮,进入“服务器配置”窗口,可以选择已有的服务账号,如图 1-21 所示。单击“对所有 SQL Server 服务使用相同的账户”按钮,选中使用的账户。SQL Server

及 SQL Server Browser 最好选为自动启动。



图 1-21 “服务器配置”窗口

(15) 单击“下一步”按钮，进入“数据库引擎配置”窗口，设置数据库登录时的身份验证。需要为 SQL Server 指定一位管理员，可以选择已有 Windows 账号，这里以系统管理员作为示例，如图 1-22 所示。如果选中混合模式，则需要输入密码。

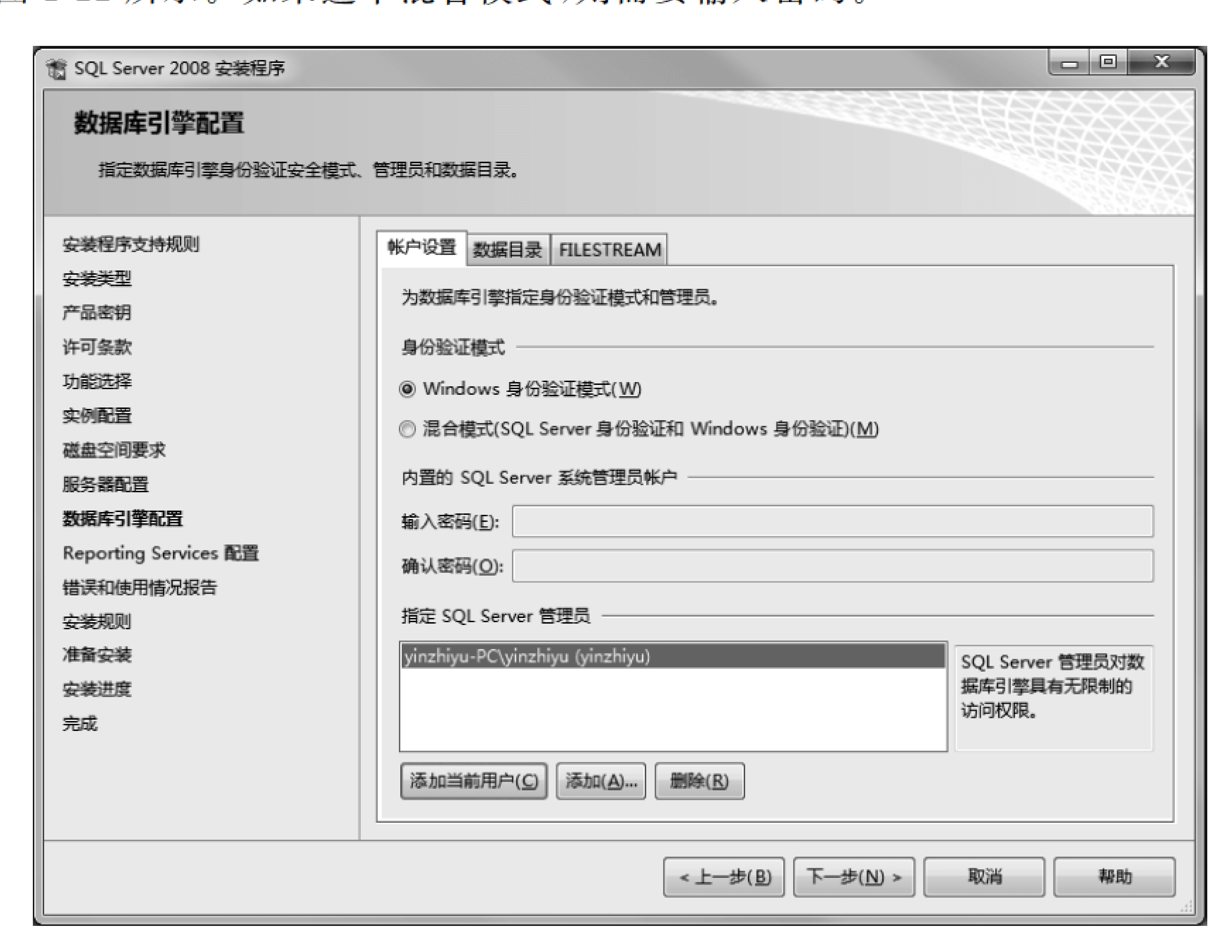


图 1-22 “数据库引擎配置”窗口

(16) 单击“下一步”按钮，进入“Reporting Services 配置”窗口，选中“安装本机模式默认配置”单选按钮，如图 1-23 所示。



图 1-23 “Reporting Services 配置”窗口

(17) 单击“下一步”按钮，进入如图 1-24 所示的“错误和使用情况报告”窗口，可选择是否将错误报告发送给 Microsoft。



图 1-24 “错误和使用情况报告”窗口

(18) 单击“下一步”按钮，进入“安装规则”窗口，安装程序根据功能配置选择，再次进行环境检查，如图 1-25 所示。

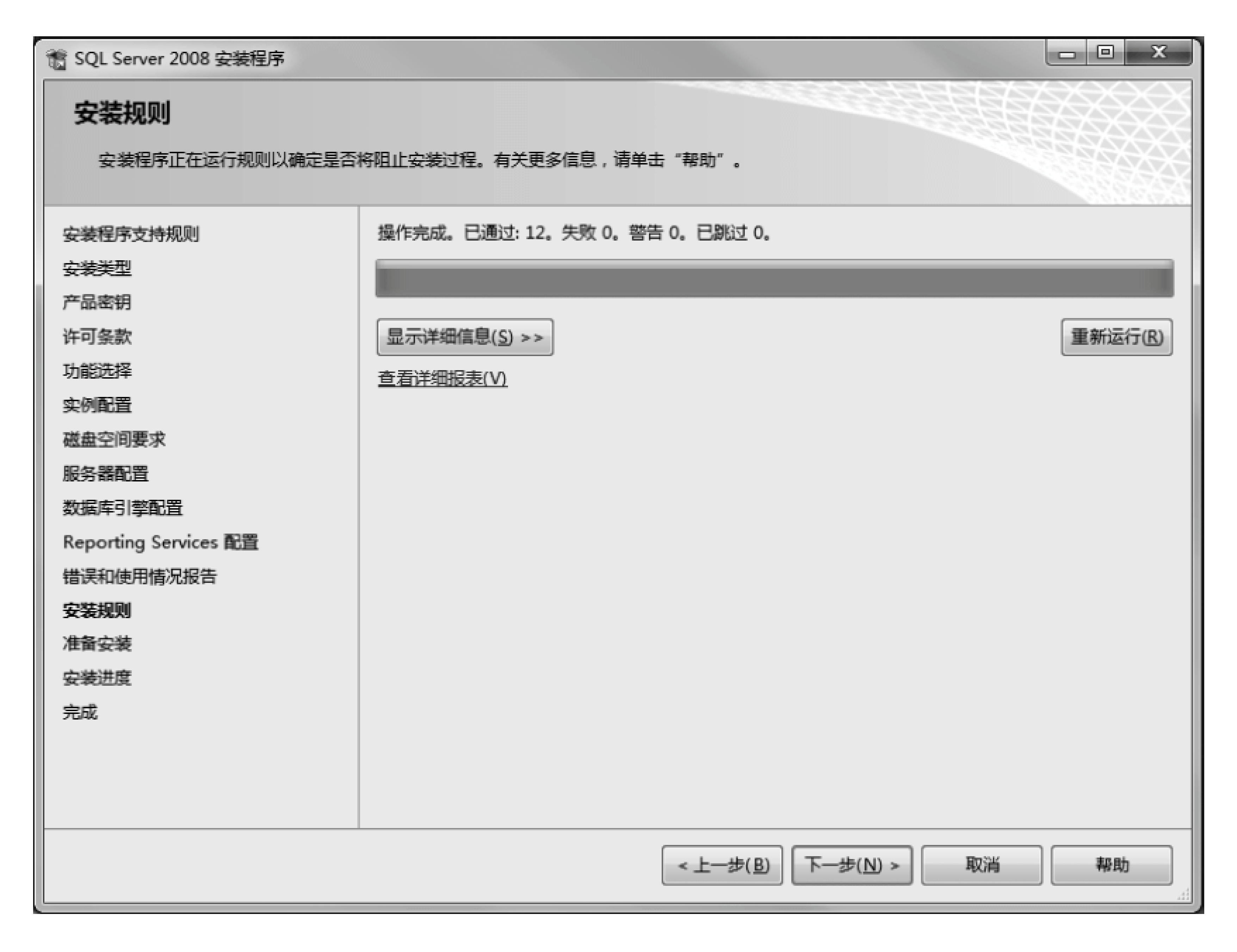


图 1-25 “安装规则”窗口

(19) 当通过检查之后，单击“下一步”按钮，进入“准备安装”窗口，软件将会列出所有的配置信息，最后一次确认安装，如图 1-26 所示。单击“安装”按钮开始 SQL Server 安装。

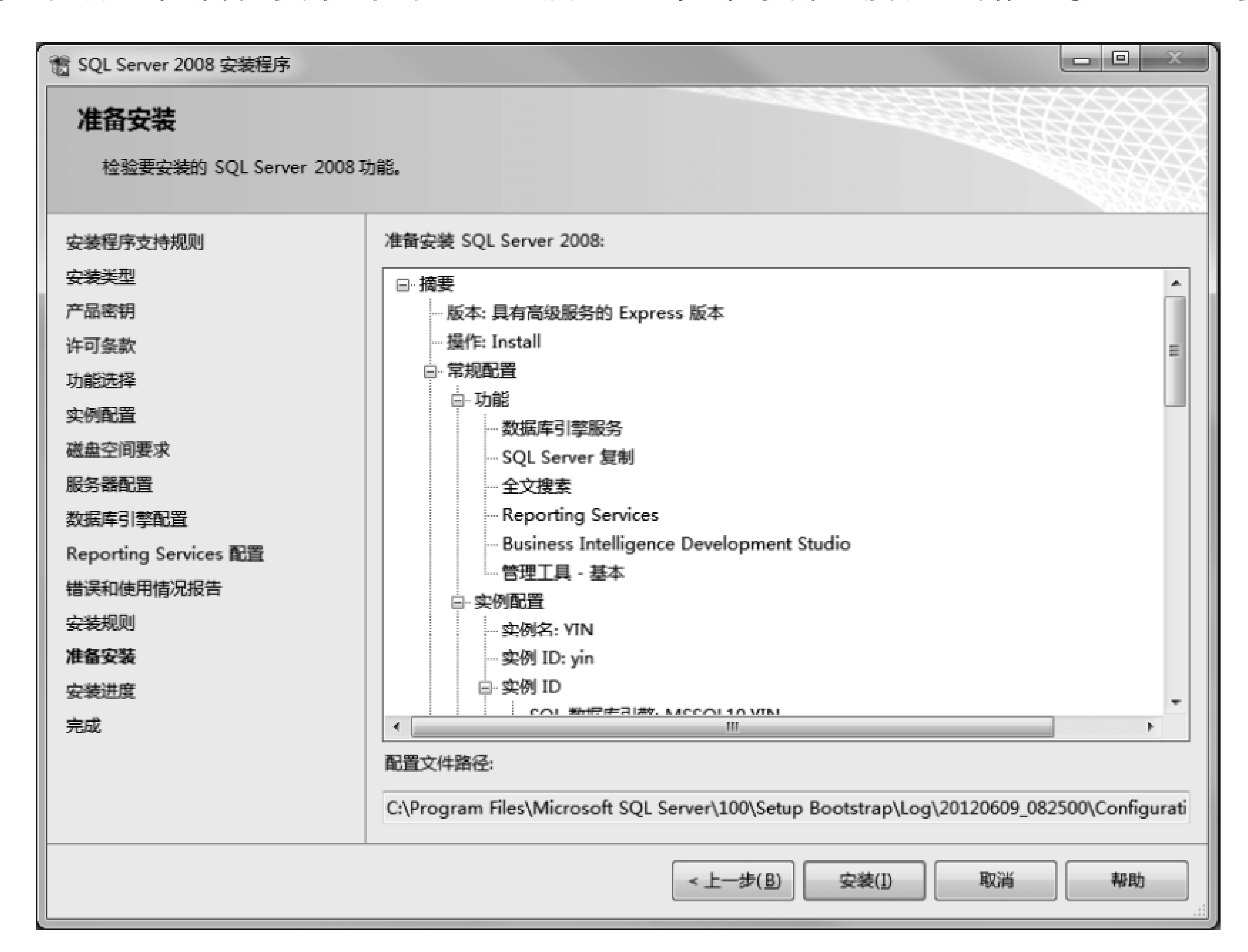


图 1-26 “准备安装”窗口

(20) 根据硬件环境的差异,安装过程可能持续10～30分钟,如图1-27所示。

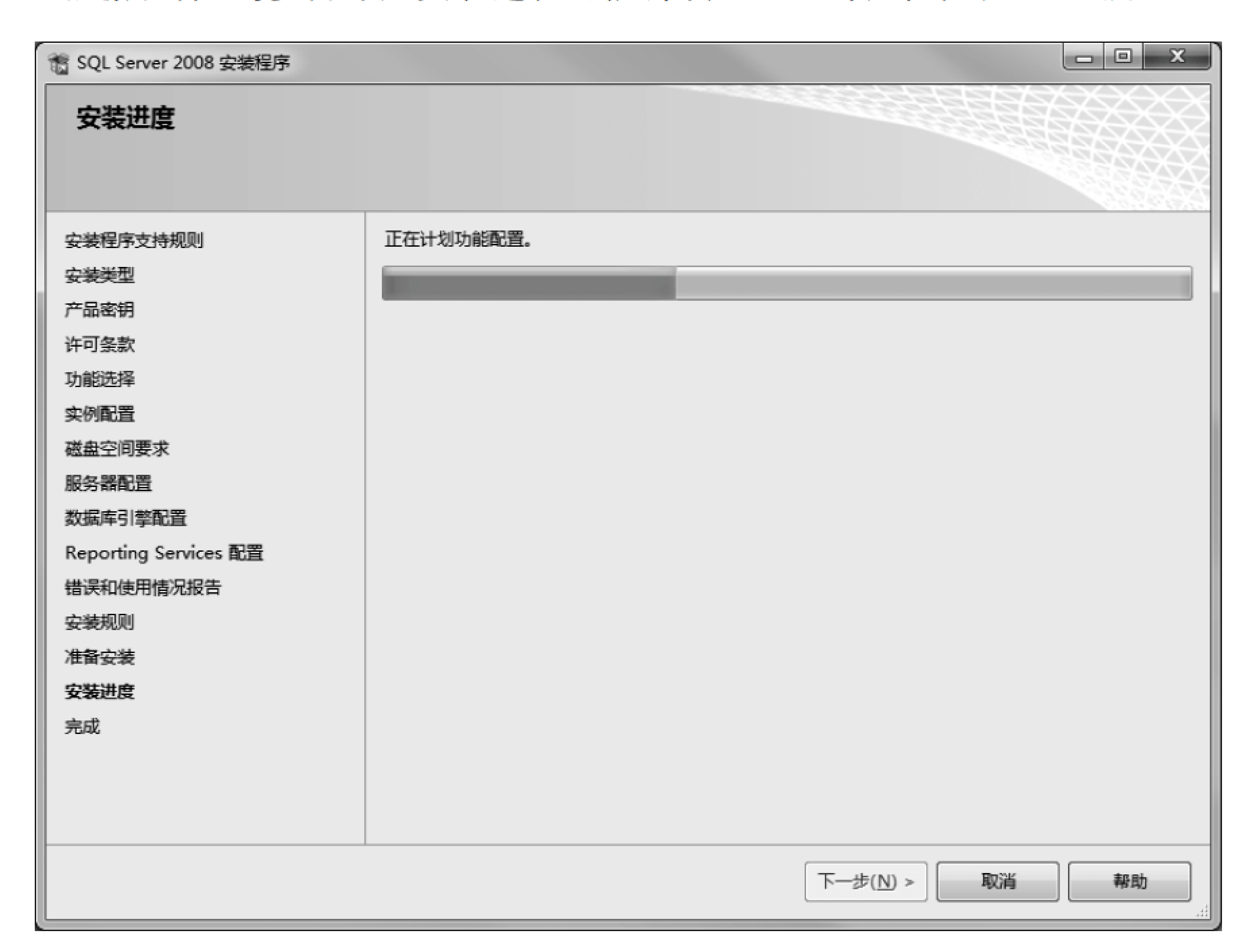


图1-27 "安装进度"窗口

(21) 如图1-28所示,当安装完成之后,SQL Server将列出各功能安装状态。

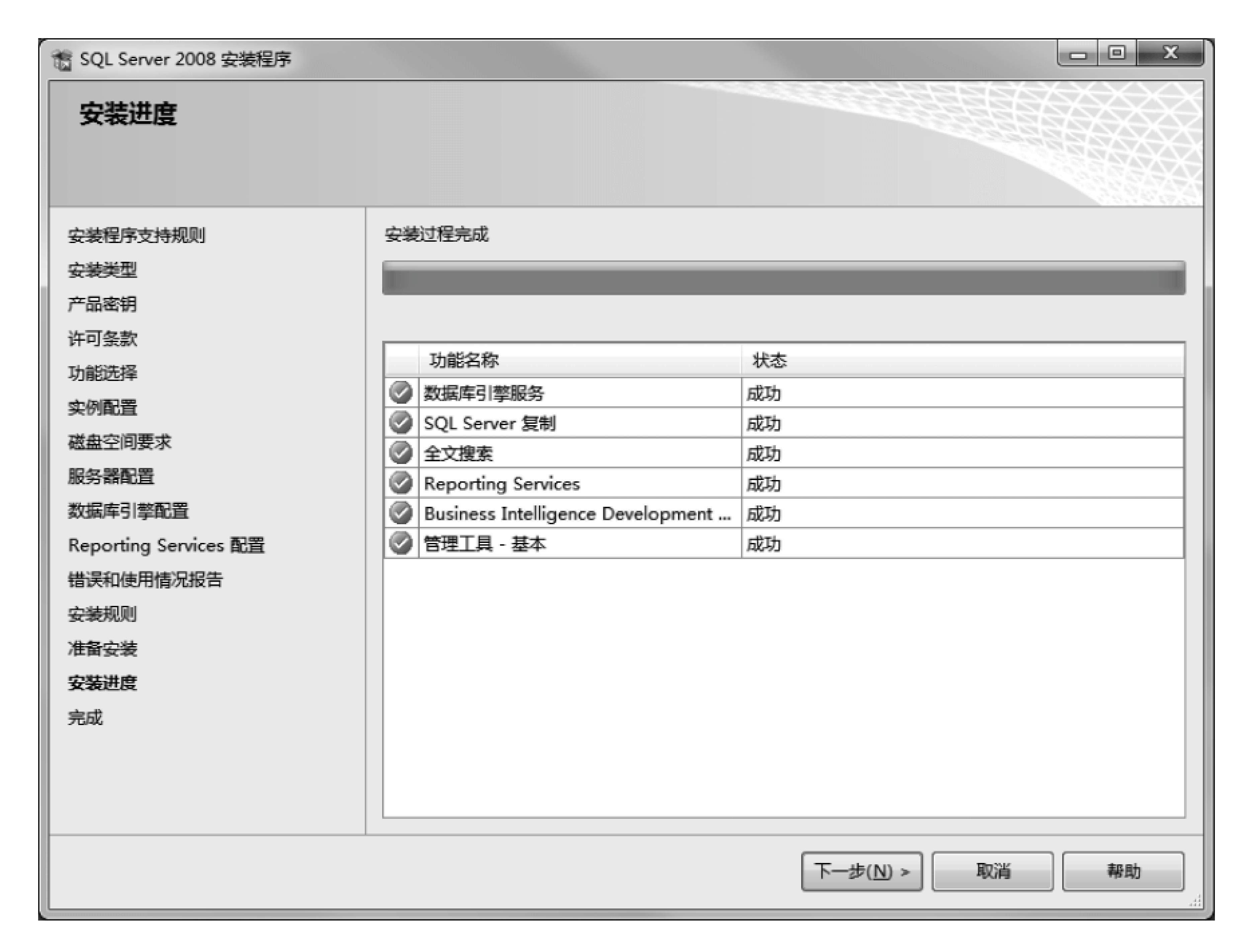


图1-28 安装过程完成

(22) 单击“下一步”按钮，进入“完成”窗口，如图 1-29 所示，此时完成了 SQL Server 2008 的安装，并将安装日志保存在了指定的路径下。

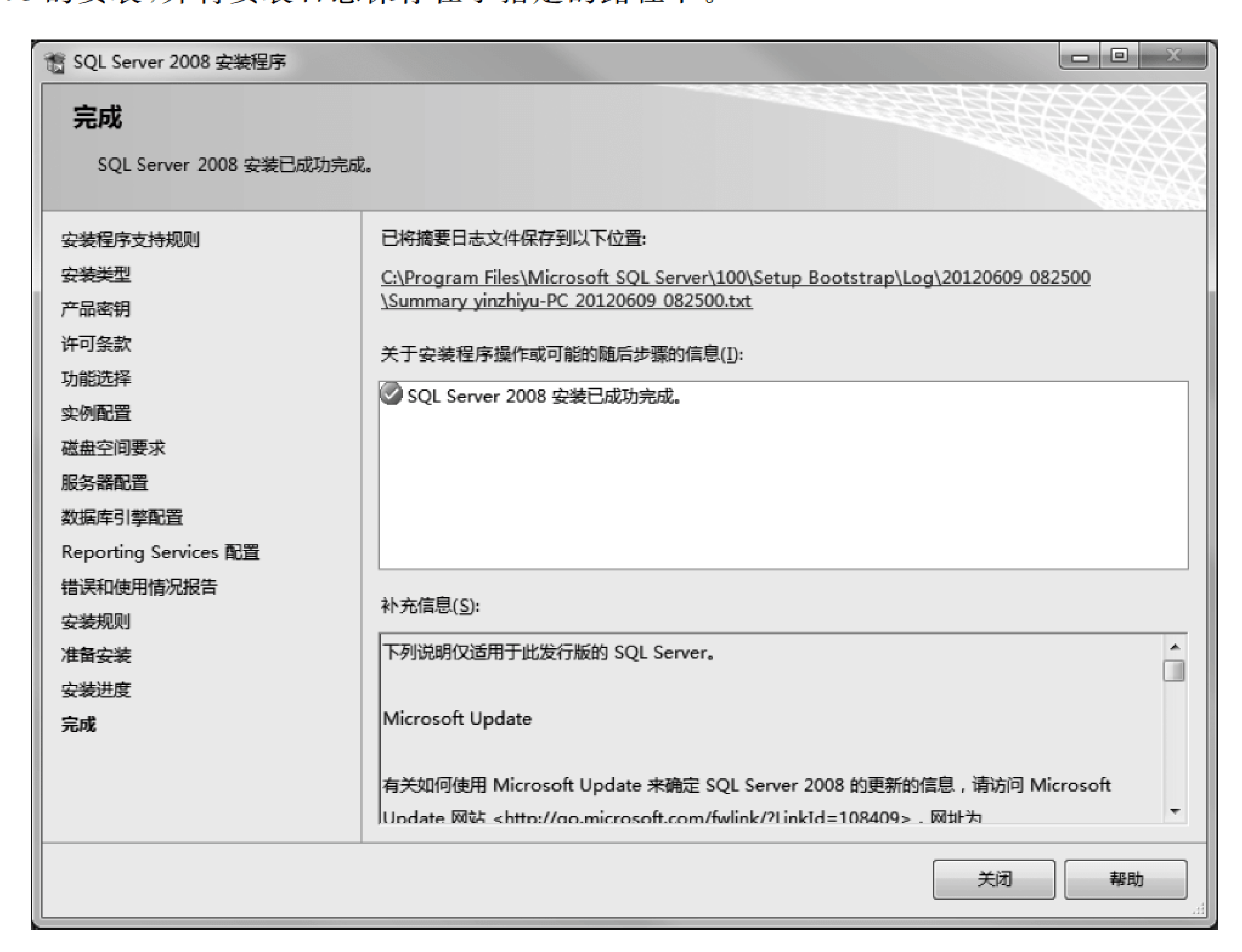


图 1-29 “完成”窗口

其他版本的产品安装过程基本相同，此处不再赘述。

1.2.6 SQL Server 2008 的主要管理工具

完成了 Microsoft SQL Server 2008 的安装后，可以使用图形化工具和命令提示实用工具进一步配置 SQL Server。下面介绍用来管理 SQL Server 2008 实例的工具。

1. 服务器管理

为了管理、配置和使用 Microsoft SQL Server 2008 系统，必须使用 Microsoft SQL Server Management Studio 工具注册服务器。

1) 注册服务器

注册服务器就是为 Microsoft SQL Server 客户/服务器系统确定一个数据库所在的机器，该机器作为服务器可以为客户端的各种请求提供服务。

在安装 SQL Server Management Studio 之后首次启动它时，将自动注册 SQL Server 的本地实例。可以使用 SQL Server Management Studio 注册其他服务器。

在 SQL Server Management Studio 的“视图”菜单中，选择“已注册的服务器”菜单项，在出现的“已注册的服务器”窗口中，右击“数据库引擎”下的 Local Server Group 文件夹，在快捷菜单中选择“新建服务器注册”命令，出现如图 1-30 所示的对话框。

在“服务器名称”下拉列表框中，既可以输入服务器名称，也可以选择一个服务器名称。



图 1-30 “新建服务器注册”对话框

从“身份验证”下拉列表框中可以选择身份验证模式，这里选择“Windows 身份验证”。

2）创建服务器组

服务器组是服务器的逻辑集合，可以利用 SQL Server Management Studio 工具把许多相关的服务器集中在一个服务器组中，方便对多服务器环境的管理操作。

在“已注册的服务器”窗口中，右击“数据库引擎”下的 Local Server Group 文件夹，在快捷菜单中选择“新建服务器组”命令，出现如图 1-31 所示的对话框。

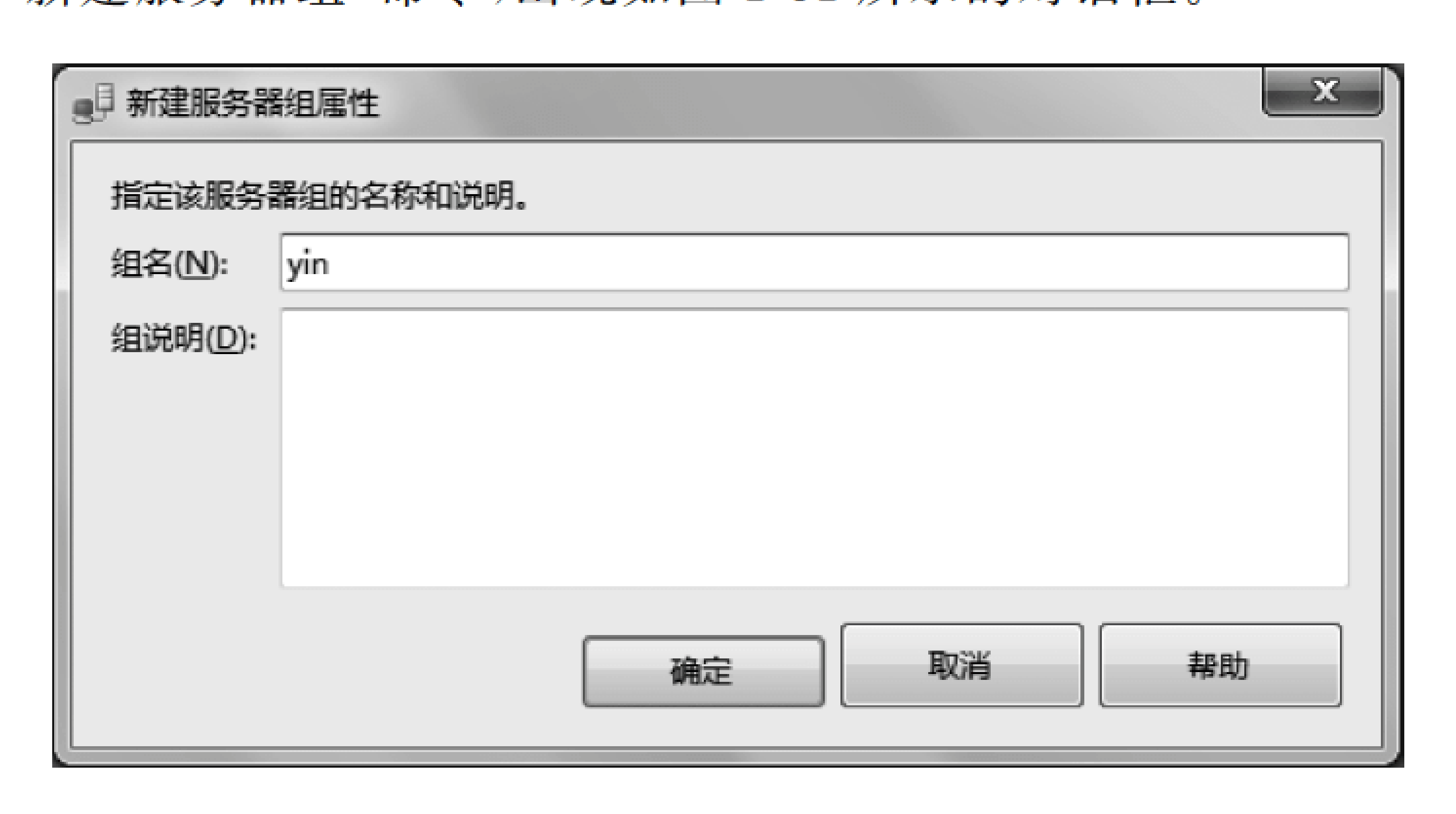


图 1-31 创建服务器组

2. SQL Server Management Studio

SQL Server Management Studio 简称 SSMS，是从 Microsoft SQL Server 2008 版本开始提供的一种新集成环境，用于访问、配置、控制、管理和开发 SQL Server 的所有组件。SQL Server Management Studio 将一组多样化的图形工具与多种功能齐全的脚本编辑器组合在一起，可为各种技术级别的开发人员和管理员提供对 SQL Server 的访问。

1）访问 SQL Server Management Studio

单击“开始”菜单，选择“所有程序”中的 SQL Server 2008 程序组，选择 SQL Server Management Studio 选项，出现如图 1-32 所示的“连接到服务器”对话框。



图 1-32 “连接到服务器”对话框

在该对话框中可以选择服务器类型、服务器名称及身份验证模式，然后单击“连接”按钮，出现如图 1-33 所示的 Microsoft SQL Server Management Studio 窗口。

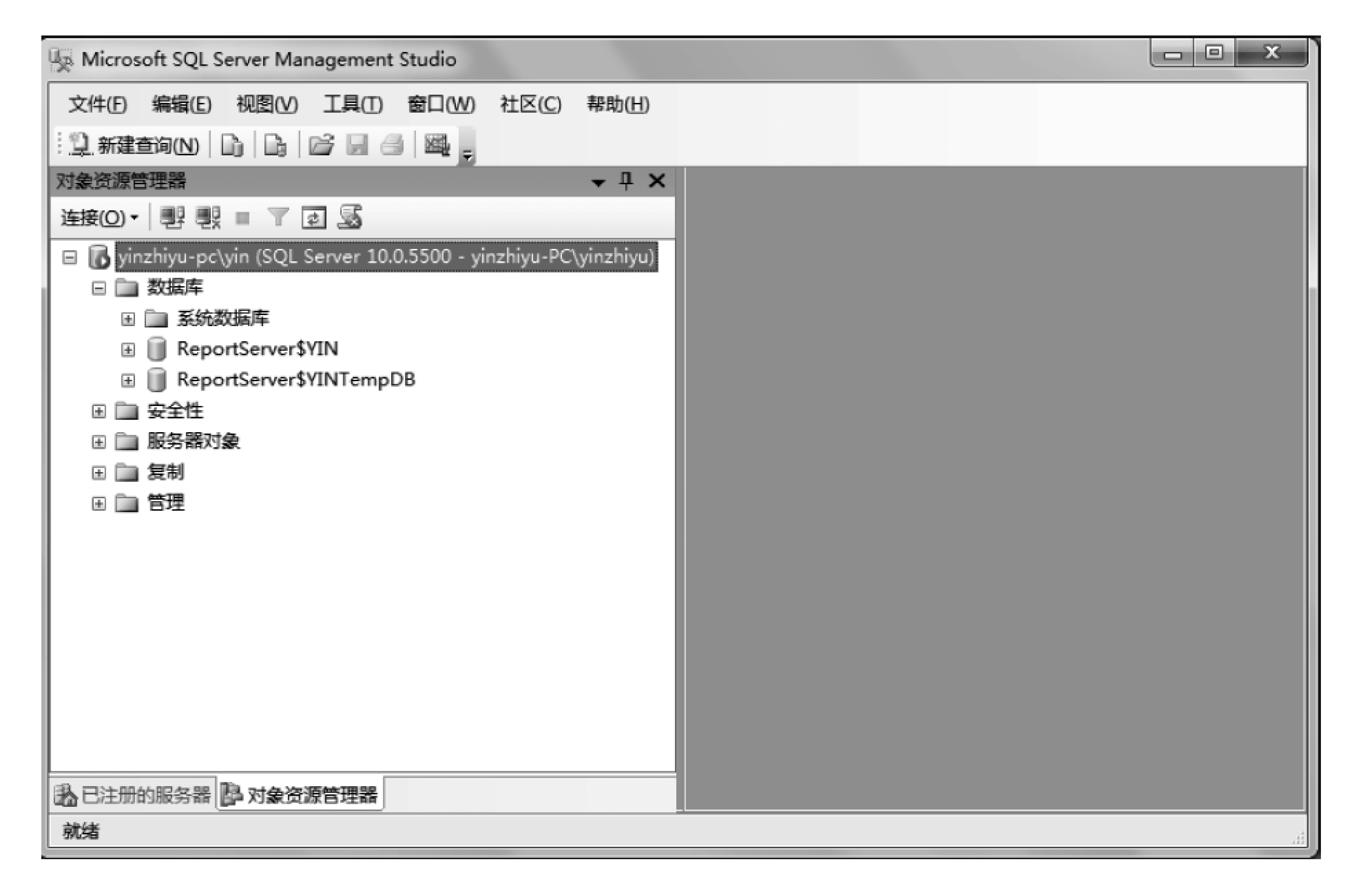


图 1-33 Microsoft SQL Server Management Studio 窗口

2）对象资源管理器

SQL Server Management Studio 的对象资源管理器组件是一种集成工具，可以查看和管理所有服务器类型的对象。

用户可以通过该组件操作数据库，包括新建、修改、删除数据库、表、视图等数据库对象，新建查询，设置关系图，设置系统安全，数据库复制，数据备份及恢复等操作，是 SQL Server Management Studio 中最常用也最重要的一个组件。

3. SQL Server 配置管理器

SQL Server 配置管理器用于管理与 SQL Server 相关联的服务，配置 SQL Server 使用的网络协议，以及从 SQL Server 客户端计算机管理网络连接配置。

使用 SQL Server 配置管理器可以启动、暂停、恢复或停止服务，还可以查看或更改服务属性。使用 SQL Server 配置管理器可以配置服务器和客户端网络协议以及连接选项。

单击“开始”菜单，选择“所有程序”中的 Microsoft SQL Server 2008 程序组，选择“配置工具”程序组中的“SQL Server 配置管理器”选项，出现如图 1-34 所示的 SQL Server 配置管理器窗口。

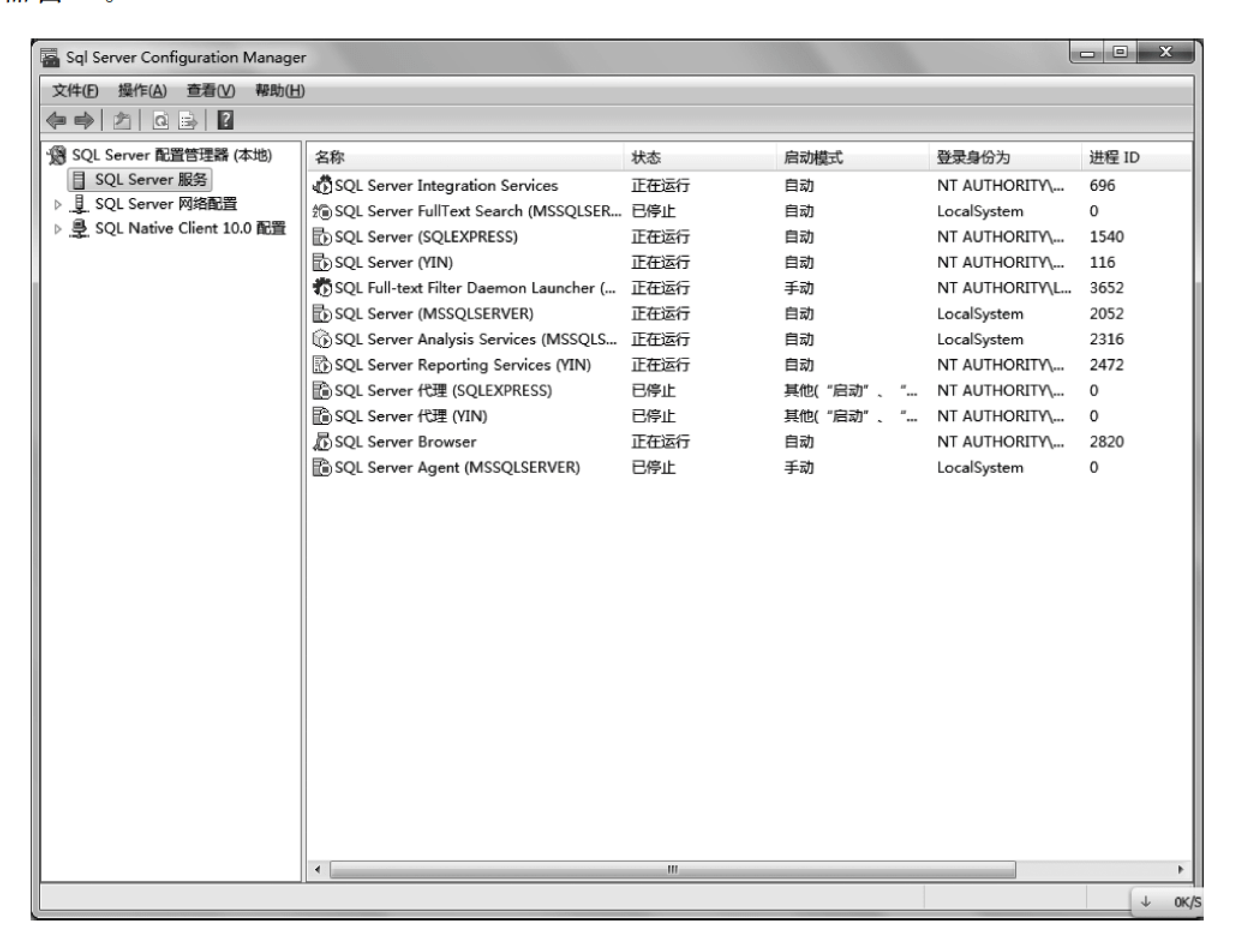


图 1-34　SQL Server 配置管理器窗口

1）管理 SQL Server 2008 服务

在 SQL Server 配置管理器窗口中，启动或停止各个服务的方法是：首先在 SQL Server 配置管理器的窗口左边单击“SQL Server 服务”，此时在窗口右边会看到已安装的所有服务，可以选中某个服务，然后单击窗口上部工具栏中的相应按钮（或右击某个服务名称，在弹出的快捷菜单中选择相应的菜单选项）来启动或停止该服务。

2）管理 SQL Server 2008 网络配置

“SQL Server 网络配置”用来配置本计算机作为服务器时允许使用的连接协议，可以启用或禁用某个协议。

当需要启用或禁用某个协议时，只需选中此协议并右击，在弹出的快捷菜单中选择“启用”或“禁用”选项即可。

注意：修改协议的状态后，还需要停止并重新启动 SQL Server 服务后，所做的更改才会生效。

3）管理 SQL Server 2008 客户端配置

SQL Server 2008 客户端配置用来配置客户端与 SQL Server 2008 服务器通信时所使用的网络协议。通过 SQL Server 2008 客户端配置工具，可以实现对客户端网络协议的启用或禁用，设置网络协议的启用顺序，并可以设置服务器别名等。

习　题　1

1. 数据库的发展历史分哪几个阶段？各有什么特点？
2. 简述数据、数据库、数据库管理系统、数据库应用系统的概念。
3. 简述数据库管理系统的功能。
4. 分析层次模型、网状模型和关系模型的特点。
5. 解释关系模型的基本概念：关系、元组、属性、域、关系模式、候选关键字、主码、外键、主属性。
6. 简述 SQL Server 的发展史。
7. 简述安装 Microsoft SQL Server 2008 系统的操作步骤。
8. SQL Server 2008 主要提供了哪些服务？如何启动、暂停或停止 SQL Server 服务？
9. 简述 SQL Server Management Studio 的使用。
10. 简述对象资源管理器的功能。

第2章 数据库的概念和操作

SQL Server 的数据库是有组织的数据的集合,这种数据集合具有逻辑结构和物理结构,并得到 DBMS 的管理和维护。数据库由包含数据的基本表和其他对象(如视图、索引、存储过程和触发器等)组成,其主要用途是处理数据管理活动产生的信息。

对数据库的操作是开发人员的一项重要工作。本章首先介绍数据库的基本概念,然后以实例的形式介绍数据库的创建、修改和删除操作。

2.1 数据库基本概念

数据库是 SQL Server 2008 存放表和索引等数据库对象的逻辑实体。数据库的存储结构分为物理存储结构和逻辑存储结构两种。

2.1.1 物理数据库

数据库的物理存储结构指的是保存数据库各种逻辑对象的物理文件是如何在磁盘上存储的,数据库在磁盘上是以文件为单位存储的,SQL Server 2008 将数据库映射为一组操作系统文件。数据库中所有的数据和对象都存储在操作系统文件中。

1. SQL Server 2008 的数据库文件的类型

SQL Server 2008 的数据库具有三种类型的文件。

(1) 主数据文件:主数据文件是数据库的起点,指向数据库中的其他文件。每个数据库都有且只有一个主数据文件。主数据文件的推荐文件扩展名是.mdf。

(2) 辅助数据文件:除主数据文件以外的所有其他数据文件都是辅助数据文件。某些数据库可能不含有任何辅助数据文件,而有些数据库则含有多个辅助数据文件。辅助数据文件的推荐文件扩展名是.ndf。

(3) 事务日志文件:日志文件包含着用于恢复数据库的所有日志信息。每个数据库必须至少有一个日志文件,当然也可以有多个。SQL Server 2008 事务日志采用提前写入的方式,即对数据库的修改先写入事务日志中,然后再写入数据库。日志文件的推荐文件扩展名是.ldf。

SQL Server 2008 不强制使用.mdf、.ndf 和.ldf 文件扩展名,但使用它们有助于标识文件的类型和用途。

在 SQL Server 2008 中,数据库中所有文件的位置都记录在该数据库的主数据文件和系统数据库 master 数据库中。

2. 数据库文件组

为了便于管理和分配数据而将文件组织在一起，通常可以为一个磁盘驱动器创建一个文件组(File Group)，将多个数据库文件集合起来形成一个整体。

SQL Server 中的数据库文件组分为主文件组(Primary File Group)和用户定义文件组(User Defined Group)。

(1) 主文件组：主文件组包含主要数据库文件和任何没有明确指派给其他文件组的其他文件。数据库的系统表都包含在主文件组中。

(2) 用户定义文件组：用户定义文件组是在 CREATE DATABASE 或 ALTER DATABASE 语句中，使用 FILEGROUP 关键字指定的文件组。

文件组应用的规则如下。

(1) 一个文件只能存在于一个文件组中，一个文件组也只能被一个数据库使用。

(2) 主文件组中包含了所有的系统表。当建立数据库时，主文件组包括主数据库文件和未指定组的其他文件。

(3) 在创建数据库对象时如果没有指定将其放在哪一个文件组中，就会将它放在默认文件组中。如果没有指定默认文件组，则主文件组为默认文件组。

(4) 事务日志文件不属于任何文件组。

2.1.2 逻辑数据库

数据库是存储数据的容器，即数据库是一个存放数据的表和支持这些数据的存储、检索、安全性和完整性的逻辑成分所组成的集合。

组成数据库的逻辑成分称为数据库对象，SQL Server 2008 中的逻辑对象主要包括数据表、视图、同义词、存储过程、函数、触发器、规则，以及用户、角色、架构等。

每个 SQL Server 都包含两种类型的数据库：系统数据库和用户数据库。

系统数据库存储有关 SQL Server 的信息，SQL Server 使用系统数据库来管理系统，例如，下面将要介绍的 master 数据库、model 数据库、msdb 数据库和 tempdb 数据库。而用户数据库由用户来建立，例如，teaching 教学库数据库。SQL Server 可以包含一个或多个用户数据库。

1. master 数据库

顾名思义，master 数据库是 SQL Server 2008 中的主数据库，它是最重要的系统数据库，记录系统中所有系统级的信息。它对其他的数据库实施管理和控制的功能，同时该数据库还保存了用于 SQL Server 管理的许多系统级信息。master 数据库记录所有的登录账户和系统配置，它始终有一个可用的最新 master 数据库备份。

由此可知，如果在计算机上安装了一个 SQL Server 系统，那么系统首先会建立一个 master 数据库来记录系统的有关登录账户、系统配置、数据库文件等初始化信息。例如，如果用户在这个 SQL Server 系统中建立一个用户数据库(如“教学库”数据库)，系统马上将用户数据库的有关用户管理、文件配置、数据库属性等信息写入 master 数据库。系统正是根据 master 数据库中的信息来管理系统和其他数据库。因此，如果 master 数据库信息被破坏，那么整个 SQL Server 系统将受到影响，用户数据库将不能被使用。

2. model 数据库

model 数据库为用户新创建的数据库提供模板，它包含了用户数据库中应该包含的所有系统表的结构。当用户创建数据库时，系统会自动把 model 数据库中的内容复制到新建的用户数据库中。用户在系统中新创建的所有数据库的内容，最初都与该模板数据库具有完全相同的内容。

3. msdb 数据库

msdb（主存）数据库记录备份及还原的历史信息、维护计划信息、作业信息、异常信息以及操作者信息等。

当很多用户在使用一个数据库时，经常会出现多个用户对同一个数据的修改而造成数据不一致的现象，或是用户对某些数据和对象的非法操作等。为了防止上述现象的发生，SQL Server 中有一套代理程序能够按照系统管理员的设定监控上述现象的发生，及时向系统管理员发出警报。那么当代理程序调度警报作业、记录操作时，系统要用到或实时产生许多相关信息，这些信息一般存储在 msdb 数据库中。

4. tempdb 数据库

使用 SQL Server 系统时，经常会产生一些临时表和临时数据库对象等，如用户在数据库中修改表的某一行数据时，在修改数据库这一事务没有被提交的情况下，系统内就会有该数据的新、旧版本之分，修改后的数据表构成了临时表。所以系统要提供一个空间来存储这些临时对象。tempdb 数据库保存所有的临时表和临时存储过程。tempdb 数据库是全局资源，所有连接到系统的用户的临时表和存储过程都被存储在该数据库中。

tempdb 数据库有个特性，即它是临时的，tempdb 数据库在 SQL Server 每次启动时都被重新创建，因此该数据库在系统启动时总是空的，上一次的临时数据都被清除掉了。临时表和存储过程在连接断开时自动清除，而且当系统关闭后将没有任何连接处于活动状态，因此 tempdb 数据库中没有任何内容会从 SQL Server 的一个启动工作保存到另一个启动工作之中。

默认情况下，在 SQL Server 运行时，tempdb 数据库会根据需要自动增长。不过，与其他数据库不同，每次启动数据库引擎时，它会重置初始大小。

master、model、msdb、tempdb 这 4 个系统数据库都是在系统安装时生成的。

2.2 数据库操作

在 SQL Server 2008 中，用户可以自己创建数据库（即用户数据库），并且可以对数据库进行修改、删除等操作。

2.2.1 创建数据库

若要创建数据库，必须确定数据库的名称、所有者、大小以及存储该数据库的文件和文件组。创建数据库时，根据数据库中预期的最大数据量，应创建尽可能大的数据文件。

在 SQL Server 2008 中创建数据库主要有两种方式：一是在 SSMS 中的界面方式下使用向导创建数据库，二是通过查询窗口执行 T-SQL 语句创建数据库。

T-SQL 全称为 Transact-Structured Query Language（事务-结构化查询语言），是标准

SQL 的扩展，在标准 SQL 的基础上增加了变量、流程控制语句等。

1. 界面方式下使用向导创建数据库

在 SSMS 中的界面方式下使用向导创建数据库的过程如下：

(1) 启动 SSMS，在对象资源管理器窗口的"数据库"节点上右击，选择快捷菜单中的"新建数据库"命令，如图 2-1 所示。



图 2-1 新建数据库

(2) 弹出"新建数据库"窗口，在"常规"选择页的"数据库名称"文本框中输入要创建的数据库的名称，如图 2-2 所示。

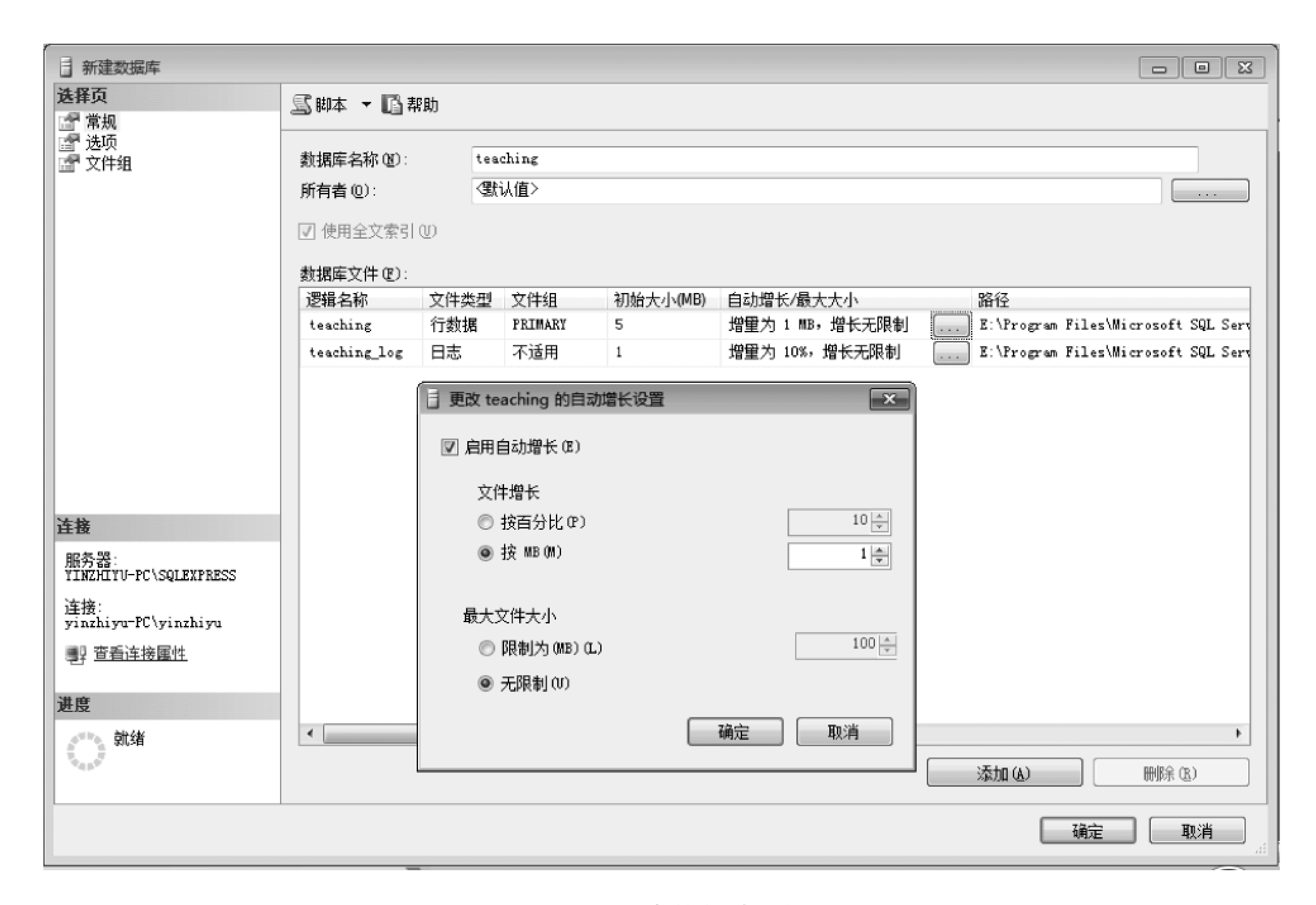


图 2-2 "新建数据库"窗口

其中，SQL Server 2008 的数据库文件拥有两个名称，即逻辑文件名和物理文件名。

① 逻辑文件名是在所有 T-SQL 语句中引用物理文件时所使用的名称。逻辑文件名必须符合 SQL Server 标识符规则，每一个数据库的逻辑文件名只有一个。

② 物理文件名是包括目录路径的物理文件名，它必须符合操作系统文件命名规则。通过上面的介绍可知，数据库中至少包含一个主数据文件和一个事务日志文件，其存储路径和文件名都可以在图 2-2 中进行修改；当然也可以利用“添加”按钮添加多个辅助数据文件和日志文件。

(3) 在“常规”选择页的数据文件“初始大小”处可以设置文件的初始大小(MB)；单击自动增长方式后的按钮可设置自动增长方式和最大文件大小；单击“路径”后的按钮可设置文件的存放路径。

(4) 在“选项”选择页中，可设置数据库的属性选项。在“文件组”选择页中，可增加或删除文件组。

在“对象资源管理器”窗口展开“数据库”选项，可以看到新建的数据库，如图 2-3 所示。



图 2-3　创建数据库成功

【例 2-1】 创建数据库 teaching(教学库)，主数据文件初始大小为 5MB，增长方式是按 10%比例自动增长；日志文件初始为 8MB，按 1MB 增长(默认是按 10%比例增长)。两个文件都不限制增长，存储位置都为“F:\DATA”。

2. 使用 T-SQL 语句创建数据库

在 SQL Server 2008 中可以利用 T-SQL 语句创建数据库。T-SQL 提供的数据库创建语句为 CREATE DATABASE，其语法格式如下：

```
CREATE DATABASE database_name
[ON [PRIMARY] [<filespec> [,…n]] [,<filegroupspec> [,…n]] ]
[LOG ON {<filespec> [,…n]}]
[FOR LOAD|FOR ATTACH]
<filespec>::= ([NAME = logical_file_name,]
FILENAME = 'os_file_name'
[,SIZE = size]
[,MAXSIZE = {max_size|UNLIMITED}]
[,FILEGROWTH = growth_increment] ) [,…n]
```

说明：在 T-SQL 语言的命令格式中，用[ ]括起来的内容表示是可选的；[,…n]表示重复前面的内容；用< >括起来的内容表示在实际编写语句时用相应的内容替代；用{ }括起来的内容表示是必选的；类似 A|B 的格式，表示 A 和 B 只能选择一个，不能同时都选。

其中的参数说明如下。

① database_name——新数据库的名称。数据库名称在服务器中必须唯一，最长为 128 个字符，并且要符合标识符的命名规则。每个服务器管理的数据库最多为 32 767 个。

② ON——指定存放数据库的数据文件信息。该关键字后面可以包含用逗号分隔的<filespec>列表，<filespec>列表用于定义主文件组的数据文件。主文件组的文件列表后可以包含用逗号分隔的<filegroupspec>列表，<filegroupspec>列表用于定义用户文件组及其中的文件。

③ PRIMARY——用于指定主文件组中的文件。主文件组不仅包含数据库系统表中的全部内容，而且还包含用户文件组中没有包含的全部对象。一个数据库只能有一个主文件，默认情况下，如果不指定 PRIMARY 关键字，则在命令中列出的第一个文件将被默认为主文件。

④ LOG ON——指明事务日志文件的明确定义。如果没有本选项，则系统会自动产生一个文件名前缀，与数据库名相同，容量为所有数据库文件大小 1/4 的事务日志文件。

⑤ NAME——指定数据库的逻辑名称，这是在 SQL Server 系统中使用的名称，是数据库在 SQL Server 中的标识符。

⑥ FILENAME——指定数据库所在文件的操作系统文件名称和路径，该操作系统文件名和 NAME 的逻辑名称一一对应。

⑦ SIZE——指定数据库的初始容量大小。如果没有指定主文件的大小，则 SQL Server 默认其与模板数据库中的主文件大小一致，其他数据库文件和事务日志文件则默认为 1MB。指定大小的数字 size 可以使用 KB、MB、GB 和 TB 后缀，默认的后缀为 MB。size 中不能使用小数，其最小值为 512KB，默认值为 1MB。主文件的 size 不能小于模板数据库中的主文件。

⑧ MAXSIZE——指定操作系统文件可以增长到的最大尺寸。如果没有指定，则文件可以不断增大直到充满磁盘。

⑨ FILEGROWTH——指定文件每次增加容量的大小，当指定数据为 0 时，表示文件不增长。增加量可以确定为以 KB、MB 作后缀的字节数或以%作后缀的被增加容量文件的百分比来表示。默认后缀为 MB。如果没有指定 FILEGROWTH，则默认值为 10%，每次扩容的最小值为 64KB。

【例 2-2】 使用 CREATE DATABASE 创建一个新的数据库，名称为 Student1，其他所有参数均取默认值。实现的步骤如下：

① 打开 SSMS，在窗口上部的工具栏的左侧找到“新建查询”按钮。

② 单击“新建查询”按钮，在 SSMS 的窗口右侧会建立一个新的查询页面，默认的名称为 SQLQuery1. sql，在这个页面中可以输入要让 SQL Server 执行的 T-SQL 语句。

③ 这里输入下面列出的创建数据库的 T-SQL 语句：

```
CREATE  DATABASE  Student1
```

④ 单击工具栏中的“执行”按钮，当系统给出的提示信息为“命令已成功完成。”时，说明此数据库创建成功，如图 2-4 所示。

【例 2-3】 创建数据库名为 Student2 的数据库，包含一个主数据文件和一个事务日志文件。主数据文件的逻辑名为 Student2_data，操作系统文件名为 Student2_data. mdf，初始容量大小为 15MB，最大容量为 20MB，文件的增长量为 20%。事务日志文件的逻辑文件名为 Student2_log，物理文件名为 Student2_log. ldf，初始容量大小为 8MB，最大容量为 12MB，文件增长量为 2MB。数据文件与事务日志文件分别存放在 E 盘和 F 盘的 DATA 文件夹下。

首先在 E 盘和 F 盘分别创建一个新的文件夹，名称是 DATA。然后，在 SSMS 窗口中

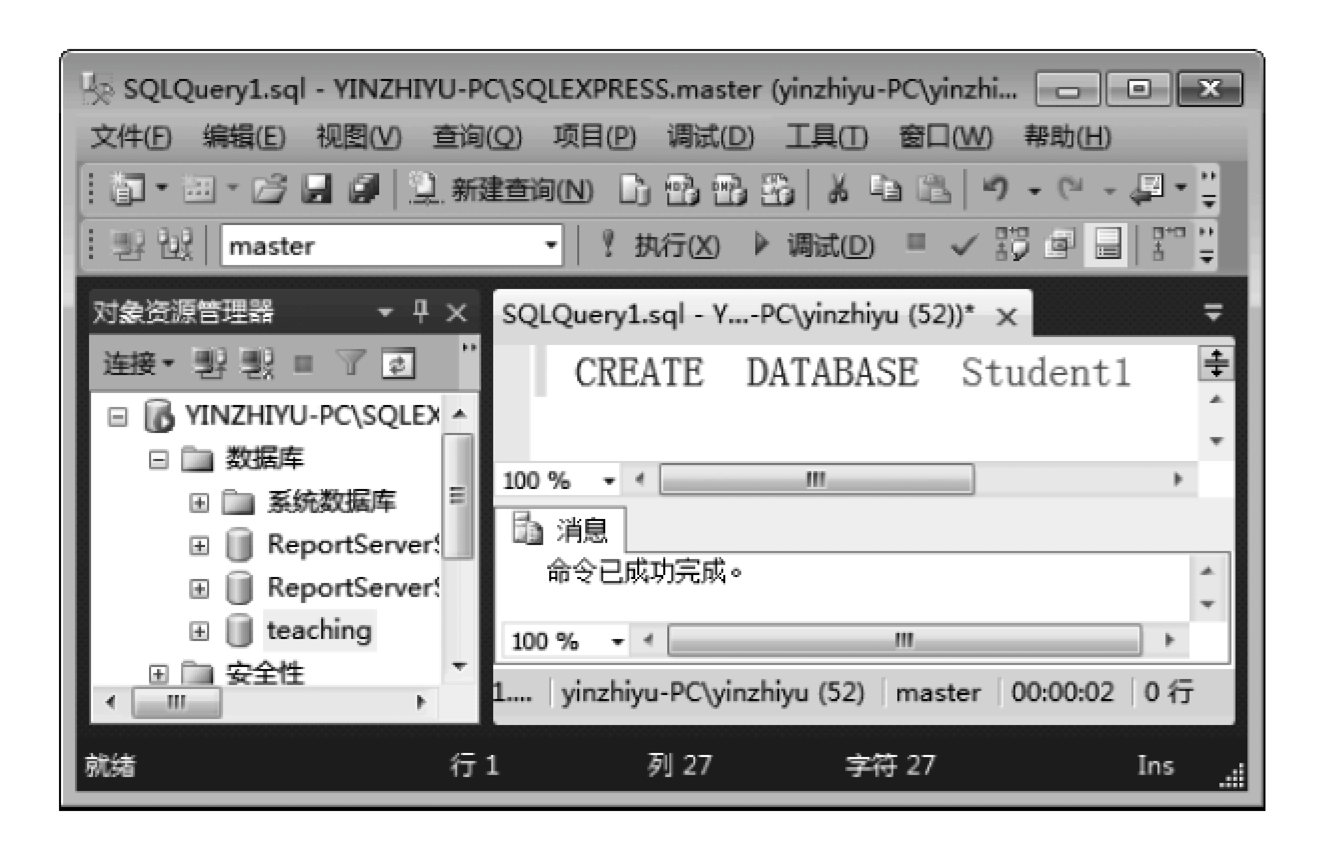


图 2-4　创建 Student1 数据库

单击“新建查询”按钮，打开其窗口，输入如下内容，单击“执行”按钮，可创建数据库。如图 2-5 所示。

```
CREATE DATABASE Student2
ON  PRIMARY
 (NAME = 'Student2_data',
FILENAME = 'E:\DATA\Student2_data.mdf',
SIZE = 15MB,
MAXSIZE = 20MB,
FILEGROWTH = 20%)
LOG ON
(NAME ='Student2_log',
FILENAME = 'F:\DATA\ Student2_log.ldf',
SIZE = 8MB,
MAXSIZE = 12MB,
FILEGROWTH = 2MB)
```

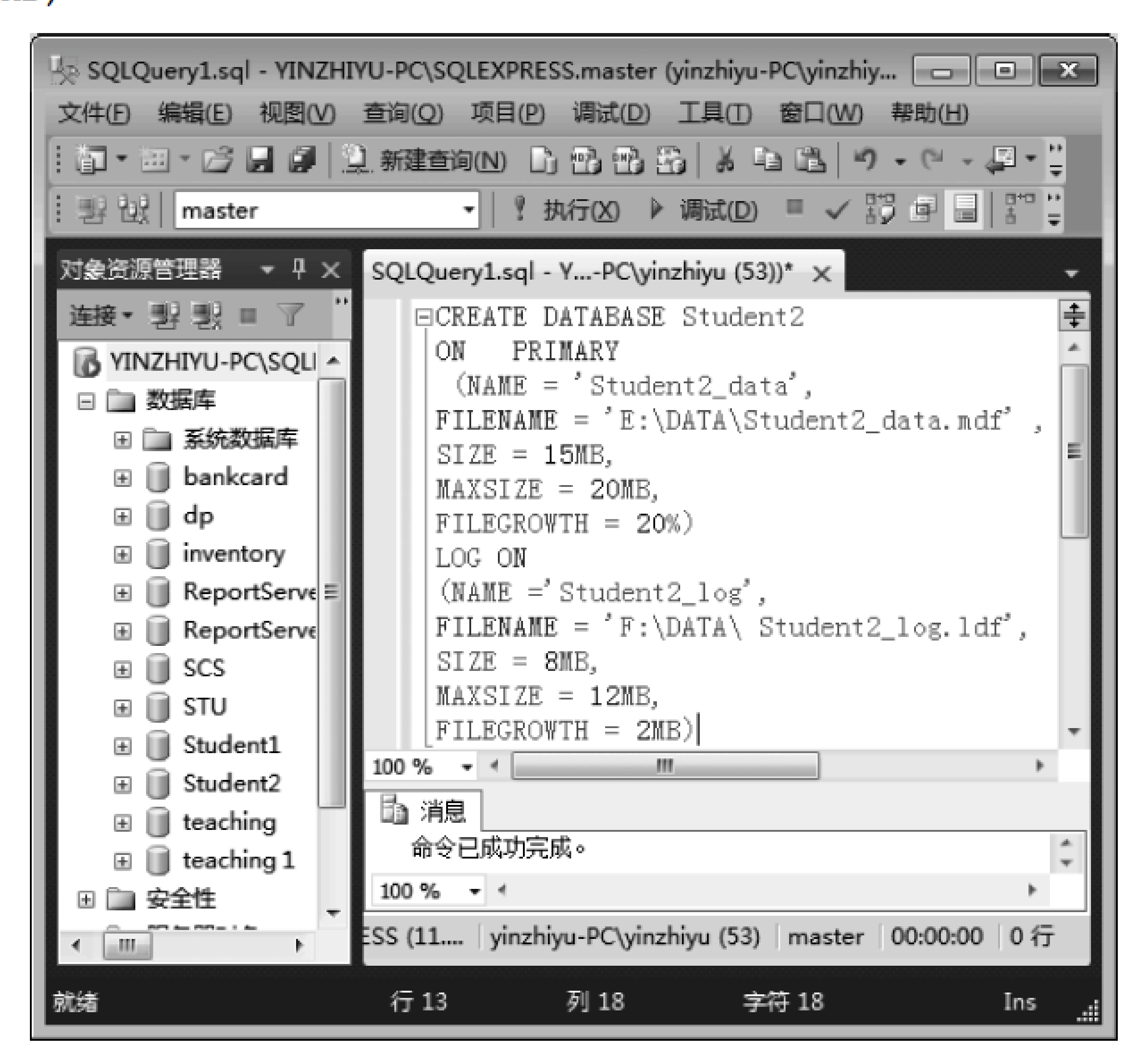


图 2-5　创建 Student2 数据库

注意：一个数据库最多可以创建 32 767 个文件组，文件组不能独立于数据库文件而建立，文件组是管理数据库中一组数据文件的管理机制。

【例 2-4】 创建一个指定多个数据文件和日志文件的数据库。该数据库名称为 Student3，有 1 个 18MB 和 1 个 20MB 的数据文件以及 2 个 10MB 的事务日志文件。数据文件逻辑名称为 Student3_1 和 Student3_2，物理文件名为 Student3_1. mdf 和 Student3_2. ndf。主数据文件 Student3_1 属于 PRIMARY 文件组，辅助数据文件 Student3_2 属于新建文件组 FG1，两个数据文件的最大大小分别为无限大和 100MB，增长速度分别为 10% 和 1MB。事务日志文件的逻辑名为 Student3_log1 和 Student3_log2，物理文件名为 Student3_log1. ldf 和 Student3_log2. ldf，最大大小均为 50MB，文件增长速度为 1MB。要求数据库文件和日志文件的物理文件都存放在 E 盘的 DATA 文件夹下。

实现的步骤如下：

首先在 E 盘创建一个新的文件夹，名称是 DATA；然后，在 SSMS 中新建一个查询页面。

输入以下程序段，并执行此查询：

```
create  database  Student3
on
(name = Student3_1,
filename = 'E:\DATA\Student3_1.mdf',
size = 18,
maxsize = unlimited,
filegrowth = 10 % ),
FILEGROUP FG1
(name =  Student3_2,
filename = 'E:\DATA\Student3_2.ndf',
size = 20,
maxsize = 100,
filegrowth = 1)
log on
(name = Student3_log1,
filename = 'E:\DATA\Student3_log1.ldf',
size = 10,
maxsize = 50,
filegrowth = 1),
(name = Student3_log2,
filename = 'E:\DATA\Student3_log2.ldf',
size = 10,
maxsize = 50,
filegrowth  = 1)
```

2.2.2 修改数据库

建好数据库后，可以对其进行修改。修改数据库包括增减数据文件和日志文件、修改文件属性(包括更改文件名和文件大小)、修改数据库选项等。

1. 增加数据库空间

1) 增加已有数据库文件的大小

在 SSMS 的“对象资源管理器”窗口中展开“数据库”选项，右击要修改的数据库的名

称，在快捷菜单中选择“属性”命令，打开数据库属性窗口，选择“文件”选择页，如图 2-6 所示，修改“初始大小”选项，修改“自动增长/最大大小”中的设置。



图 2-6 “数据库属性”窗口

也可以使用 T-SQL 语句增加已有数据库文件的大小，语法格式如下：

```
ALTER DATABASE 数据库名
MODIFY FILE
{<filespec> [,...n]}
```

【例 2-5】 为 Student2 数据库增加容量，原来数据库文件 Student2_data 的初始分配空间为 15MB，现在将 Student2_data 的分配空间增加至 20MB。

```
ALTER DATABASE Student2
MODIFY FILE
 (NAME = Student2_data,
  SIZE = 20MB)
```

2）增加数据库文件

在 SSMS 中，选择“数据库属性”窗口中的“文件”选择页，单击“添加”按钮，为新的数据库文件指定逻辑文件名、初始大小、文件增长方式等属性，再单击“确定”按钮即可完成增加数据库文件数目的操作。

也可以使用 T-SQL 语句增加数据库文件的数目，语法格式如下：

```
ALTER DATABASE 数据库名
```

```
ADD FILE|ADD LOG FILE
{<filespec>[,…n] | [<filegroupspec>[,…n]]  ]}
```

【例 2-6】 为数据库 Student2 增加数据文件 Student2_data1，初始大小 10MB，最大 50MB，文件增长的量为 5%。

```
ALTER DATABASE Student2
ADD FILE
( NAME = 'Student2_data1',
  FILENAME = 'F:\DATA\ Student2_data1.ndf',
  SIZE = 10MB,
  MAXSIZE = 50MB,
  FILEGROWTH = 5%)
```

2. 缩减已有数据库文件的大小

数据库文件大小可以缩减，具体步骤如下：

① 在 SSMS 的“对象资源管理器”窗口的数据库名上右击，选择快捷菜单中的“任务”→“收缩”→“数据库”命令，在出现的“收缩数据库”对话框中，保持默认设置，单击“确定”按钮，数据库收缩完毕。

② 如果要收缩特定的数据文件或日志文件，则选择快捷菜单中的“任务”→“收缩”→“文件”命令。

③ 数据库的自动收缩可以在数据库属性窗口的“选项”选择页面中设置，只要将选项中的“自动收缩”设为 True 即可。

注意： 为了避免存储空间的浪费，可以进行数据库的手动收缩或设置自动收缩。但是，无论怎么收缩，数据库的大小也不会小于其初始大小，所以创建数据库时初始大小的选择应尽可能合理。

3. 删除数据库文件

在 SSMS 中选择数据库属性窗口中的“文件”选择页，指定要删除的文件，单击“删除”按钮就可以删除对应的文件，从而缩减了数据库的空间。

使用 ALTER DATABASE 的 REMOVE FILE 子句，可以删除指定的文件。语法格式如下：

```
ALTER DATABASE 数据库名
REMOVE FILE 逻辑文件名
```

【例 2-7】 将数据库 Student2 中增加的数据文件 Student2_data1 删除。

```
ALTER DATABASE Student2
REMOVE FILE Student2_data1
```

4. 数据库更名

数据库建好后，可以更改其名称。在重命名数据库之前，应该确保没有用户正在使用该数据库。

常用更名方法有两种：

方法一，在 SSMS 中选中此数据库，右击，在弹出的快捷菜单中选择“重命名”命令。

方法二，在查询窗口执行系统存储过程 sp_renamedb 更改数据库的名称。系统存储过

程 sp_renamedb 语法如下：

```
sp_renamedb [@dbname = ]'old_name',[@newname = ]'new_name'
```

【例 2-8】 将已存在的数据库 Student2 重命名为 Student_back。

```
sp_renamedb  'Student2', 'Student_back'
```

2.2.3 删除数据库

不再使用的数据库可以删除，删除数据库的方法有如下两种。

1）使用 SSMS 删除数据库

打开 SSMS，选择“数据库”选项，右击要删除的数据库，在弹出的快捷菜单中选择“删除”命令，在随后出现的“删除对象”对话框中单击“确定”按钮，即可完成对指定数据库的删除操作。

2）使用 T-SQL 中的 DROP DATABASE 语句删除数据库

语法格式如下：

```
DROP  DATABASE 数据库名
```

【例 2-9】 删除已创建的数据库 Student3。

```
drop database Student3
```

说明：用户只能根据自己的权限删除用户数据库；不能删除当前正在使用（正打开供用户读写）的数据库；无法删除系统数据库（msdb、model、master、tempdb）。

习 题 2

1. 简述数据库的两种存储结构。
2. 数据库由哪几种类型的文件组成？其扩展名分别是什么？
3. 简述 SQL Server 2008 中文件组的作用和分类。
4. 使用 SSMS 创建名为 bankcard（银行卡管理）的数据库，并设置数据库主文件名为 bankcard_data，初始大小为 10MB；日志文件名为 bankcard_log，初始大小为 10MB，其他默认。所有的文件都放在目录“F:\DATA”中。
5. 删除习题 4 创建的数据库，使用 T-SQL 语句再次创建该数据库，主文件和日志文件的文件名、初始大小及存放位置同上，要求：bankcard_data 最大为无限大，增长速度为 20%，bankcard_log 最大为 50MB，增长速度为 1MB。
6. 请分别使用 SSMS 和 T-SQL 语句创建数据库 Student，要创建的数据库的要求如下所示：数据库名称为 student，包含 3 个 20MB 的数据库文件，2 个 10MB 的日志文件，创建使用一个自定义文件组，所有名称自定，主文件为第一个文件，主文件的扩展名为.mdf，次要文件的扩展名为.ndf；要明确地定义日志文件，日志文件的扩展名为.ldf；自定义文件组包含后两个数据文件，所有的文件都放在目录“E:\DATA”中。

第3章　表 的 操 作

在数据库中，表是由数据按一定的顺序和格式构成的数据集合，是存放数据的基本单位，是数据库的主要对象。表的数据组织形式是行、列结构，表中每一行代表一条记录，每一列代表记录的一个字段，没有记录的表称为空表。每个表通常都有一个主关键字(又称为主码)，用于唯一地确定一条记录。在同一个表中不允许有相同名称的字段。

本节将以在 teaching 数据库中表的操作为例，介绍表的基本操作，包括表的创建、修改和删除操作，表中数据的插入、修改、删除操作等内容。

3.1　创　建　表

创建好数据库后，数据库是空的，逻辑上就像建造了一个空的房子(仓库)，物理上是创建了几个操作系统文件(数据文件和日志文件)。存入数据后，才成为真正的数据库。对于关系数据库而言，用于存储数据的当然是关系表，所以首先要在空数据库中创建表。

注意：表必须建在某一数据库中，不能单独存在，也不能以操作系统文件形式存在。

3.1.1　数据类型

当定义数据表的字段、声明程序中的变量等的时候，都需要为它们设置一个数据类型，目的是指定该字段或变量所存放的数据是整数、字符串、货币、日期时间或是其他类型的数据，以及会用多少空间来存储数据。

数据类型决定了数据的存储格式，代表了各种不同的信息类型。SQL Server 提供系统数据类型集，该类型集定义了可与 SQL Server 一起使用的所有数据类型。

SQL Server 中的数据类型可分为系统内置数据类型和用户自定义数据类型两种。系统数据类型是 SQL Server 预先定义好的，可以直接使用。

1. ASCII 字符型

ASCII 字符数据的类型包括 char、varchar 和 text。ASCII 字符数据是由任何英文字母、符号、数字以及中国编码标准的汉字任意组合而成的数据，每个英文字母、符号或数字占用一个字节，每个汉字占用两个字节。

① char(n)按固定长度存储字符串，字符数不满 n 个时，自动补空格，n 的取值为 1～8000 的整数。因为每个汉字占用两个字节，所以当用此类型存储汉字时，n 表示字节数。

② varchar(n)按变长存储字符串，存储大小为输入数据的字节的实际长度，若输入的数据超过 n 个字节，则截断后存储。n 的取值同样为 1～8000 的整数。char 类型的字符串查询速度快，但为了节省存储空间，当有空值或字符串数据长度不固定时可以使用 varchar 数

据类型。

③ text 数据类型可以存储最大长度为 $2^{31}-1$ 字节的字符数据。超过 8KB 的 ASCII 数据可以使用 text 数据类型存储。

2. Unicode 字符型

Unicode(统一编码)为国际通用字符类型,该类型包括 nchar、nvarchar 和 ntext。Unicode 字符数据是由任何英文字母、符号、数字以及国际标准的汉字、韩文、日文等任意组合而成的数据,每个字符都占用 2 字节。

① nchar(n)存放固定长度的 n 个 Unicode 字符数据,n 必须是一个 1～4000 的整数。

② nvarchar(n)存放长度可变的 n 个 Unicode 字符数据,n 必须是一个 1～4000 的整数。

③ ntext 存储最大长度为 $2^{30}-1$ 字节的 Unicode 字符数据。

3. 整型

① bigint(大整数):-2^{63}～$2^{63}-1$ 的整型数据即所有数字。存储大小为 8B。

② int(整型):-2^{31}～$2^{31}-1$ 的整型数据所有数字。存储大小为 4B。

③ smallint(短整型):-2^{15}～$2^{15}-1$ 的整型数据。存储大小为 2B。

④ tinyint(微短整型):0～255 的整型数据。存储大小为 1B。

⑤ bit(位):只存储 null、0 或 1,只占据 1B 空间。bit 数据类型非常适合用于开关标记,在大多数应用程序中被转换为 true 或 false。

4. 精确数值型

精确数值型数据由整数部分和小数部分构成,其所有的数字都是有效位,能够以完整的精度存储十进制数。

在 SQL Server 中精确数值型是 decimal 和 numeric,两者唯一的区别在于 decimal 不能用于带有 identity 关键字的列。

表达方式:decimal[(p[, s])]和 numeric[(p[, s])]。

其中,p:指定精度或对象能够控制的数字个数,s:指定可放到小数点右边的小数位数或数字个数;p 可指定的范围为 1～38,s 可指定的范围最少为 0,最多不可超过 p。

decimal(8,6)取值范围是-99.999999～99.999999。

5. 近似数值型

① float[(n)]:存放-1.79E+308～1.79E+308 数值范围内的浮点数,其中 n 为精度(尾数的位数),n 是 1～53 的整数。SQL Server 对此只使用两个值:如果指定位于 1～24,就使用 24,存储大小为 4B;如果指定 25～53,就使用 53,存储大小为 8B。

② real:-3.40E + 38～3.40E + 38 的浮点数字数据,存储大小为 4B。real 与 float(24)作用相同。

6. 日期时间型

① datetime 数据类型可以存储从 1753 年 1 月 1 日到 9999 年 12 月 31 日的日期和时间数据,每个日期时间型数据都需要 8 个存储字节,精确度为千分之三秒,时间范围为 00:00:00:000 到 23:59:59.999。

② smalldatetime 数据类型可以存储从 1900 年 1 月 1 日到 2079 年 6 月 6 日的日期和时间数据，每个小日期时间型数据都需要 4 个存储字节，精确度为分，时间范围为 00:00 到 23:59。

使用旧的日期时间数据类型时，SQL Server 用户无法分别处理日期和时间信息。SQL Server 2008 以上版本新增的四种新数据类型(date、time、datetime2 和 datetimeoffset)则改变了这一状况，从而简化了日期和时间数据的处理，并且提供了更大的日期范围、小数秒精度以及时区支持。新数据库应用程序应使用这些新数据类型，而非原来的 datetime。

③ date 数据类型仅存储日期，不存储时间。范围是从公元元年 1 月 1 日到 9999 年 12 月 31 日。每个日期型数据都需要 3 个存储字节，且精度为 10 位。date 类型的准确性仅限于单天。

④ time[(n)]数据类型仅存储一天中的时间，不存储日期。它使用的是 24 小时时钟，因此支持的范围是 00:00:00.0000000 到 23:59:59.9999999(小时、分钟、秒和小数秒)。可在创建数据类型时指定小数秒的精度，即 n 的值，默认精度是 7 位，准确度是 100 毫微秒。精度影响着所需的存储空间大小，范围包括最多 2 位的 3 个字节、3 或 4 位的 4 个字节以及 5～7 位的 5 个字节。

⑤ datetimeoffset[(n)]数据类型提供了时区信息。time 数据类型不包含时区，因此仅适用于当地时间。然而，在全球经济形势下，常常需要知道某个地区的时间与另一地区的时间之间的关系。范围是从公元元年 1 月 1 日 00:00:00.0000000 到 9999 年 12 月 31 日 23:59:59.9999999。可在创建数据类型时指定小数秒的精度，即 n 的值，默认精度是 7 位。

⑥ datetime2[(n)]数据类型是原始 datetime 类型的扩展。它支持更大的日期范围以及更细节的小数秒精度，同时可使用它来指定精度。datetime2 类型的日期范围是公元元年 1 月 1 日到 9999 年 12 月 31 日(原始 datetime 的范围则是 1753 年 1 月 1 日到 9999 年 12 月 31 日)。与 time 类型一样，提供了 7 位小数秒精度，时间范围为 00:00:00.0000000 到 23:59:59.9999999。

日期的格式可以设定。设置日期格式的命令如下：

```
Set DateFormat {format | @format _var}
```

其中，format | @format_var 是日期的顺序。有效的参数包括 MDY、DMY、YMD、YDM、MYD 和 DYM。在默认情况下，日期格式为 MDY。

注：该设置仅用在将字符串转换为日期值时的解释中，它对日期的显示没有影响。

SQL Server 中常用的日期和时间表示格式：

① 分隔符可用“/”“-”或“.”，例如，'4/15/2008'、'4-15-05'或'4.15.2008'。

② 字母日期格式：'April 15,2008'。

③ 不用分隔符：'20080501'。

④ 时:分:秒:毫秒：'08:05:25:28'。

⑤ 时:分 AM|PM：'05:08AM'、'08:05PM'。

7. 货币型

① money：货币数据值为 $-2^{63}\sim2^{63}-1$，精确到货币单位的万分之一。存储大小

为 8B。

② smallmoney：货币数据值为－214 748.3648～＋214 748.3647，也可以精确到货币单位的万分之一。存储大小为 4B。

8. 二进制类型

① binary[(n)]为存储空间固定的数据类型，存储空间大小为 n＋4B。n 必须为 1～8000。若输入的数据不足 n＋4B，则补足后存储。若输入的数据超过 n＋4B，则截断后存储。

② varbinary[(n)]按变长存储二进制数据。n 必须为 1～8000。若输入的数据不足 n＋4B 时，则按实际数据长度存储。若输入的数据超过 n＋4B，则截断后存储。binary 数据比 varbinary 数据存取速度快，但是浪费存储空间，用户在建立表时，选择哪种二进制数据类型可根据具体的使用环境来决定。若不指定 n 的值，则默认为 1。

③ image 数据类型可以存储最大长度为 $2^{31}-1$ 字节的二进制数据。

9. 其他数据类型

除了前面介绍的数据类型之外，Microsoft SQL Server 系统还提供了 CURSOR、SQL_VARIANT、TABLE、TIMESTAMP、UNIQUEIDENTIFIER 及 XML 等数据类型。

MicrosoftSQL Server 数据类型如表 3-1 所示。

表 3-1 T-SQL 的基本数据类型

数 据 类 型	符 号 标 识
整型	bigint、int、smallint、tinyint、bit
精确数值型	decimal、numeric
浮点型	float、real
货币型	money、smallmoney
字符型	char、varchar
Unicode 字符型	nchar、nvarchar、text、ntext
图像型	image
二进制型	binary、varbinary
日期时间型	datetime、smalldatetime、date、time、datetime2、datetimeoffset
特殊数据类型	cursor、hierarchyid、timestamp、sql_variant、table、uniqueidentifier、XML

3.1.2 界面方式创建表

对于具体的某一个表，在创建之前，需要确定表的下列特征。

① 表要包含的数据；

② 表中的列数，每一列中数据的类型和长度（除 ASCII、Unicode 字符型数据和二进制类型数据可以根据需要设置外，其他数据类型的长度均默认），哪些列允许空值；

③ 是否要使用以及何处使用约束、默认设置和规则；

④ 所需索引的类型，哪里需要索引，哪些列是主码，哪些列是外键。

【例 3-1】 在数据库 teaching 中创建 student 学生表，student 表的表结构定义如表 3-2 所示。

表 3-2　student 表结构

列名	数据类型	长度	允许空值	键值	取 值 范 围	含义
sno	char	9	否	主码		学号
sname	nvarchar	8	否			姓名
ssex	nchar	1	否		男或女	性别
birthday	date		是			出生日期
en_time	date		是		默认值：当前系统日期	入学时间
specialty	nvarchar	10	是			专业
grade	nchar	5	否			年级

创建步骤如下。

(1) 打开 SSMS，在对象资源管理器中，右击 teaching 数据库的"表"节点，选择"新建表"命令，如图 3-1 所示。

(2) 出现表设计器窗口，在其上半部分输入列的基本属性。在其下半部分的列属性中指定列的详细属性。

(3) 选中要设置为主码的列(sno)，单击工具栏上的钥匙形按钮，或选择"表设计器"菜单中的"设置主码"命令将其设为主码，如图 3-2 所示。

(4) 右击性别列(ssex)，在弹出的快捷菜单中选择"CHECK 约束(O)"选项，打开"CHECK 约束"对话框，单击"添加"按钮，在"表达式"文本框中输入取值范围逻辑表达式，如图 3-3 所示，单击"关闭"按钮；选中入学时间列(en_time)，在下半部分详细属性窗口的"默认值或绑定"项中输入默认值表达式，这里为系统函数 getdate()，如图 3-3 所示。

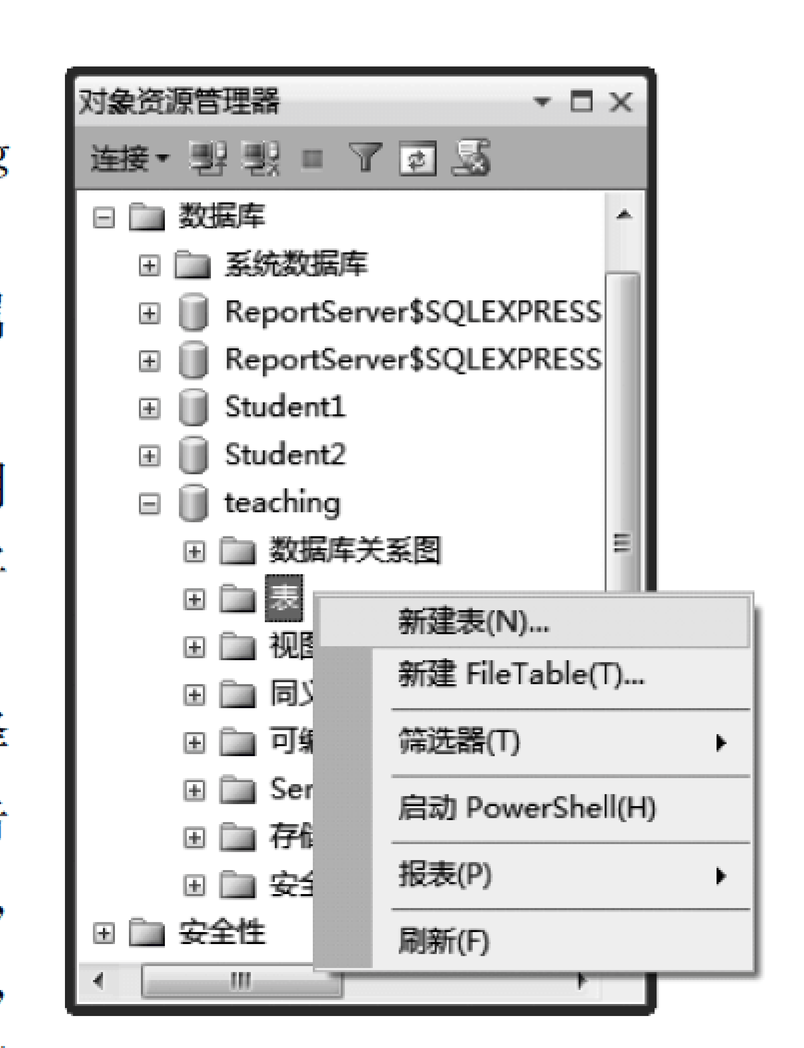


图 3-1　新建表菜单

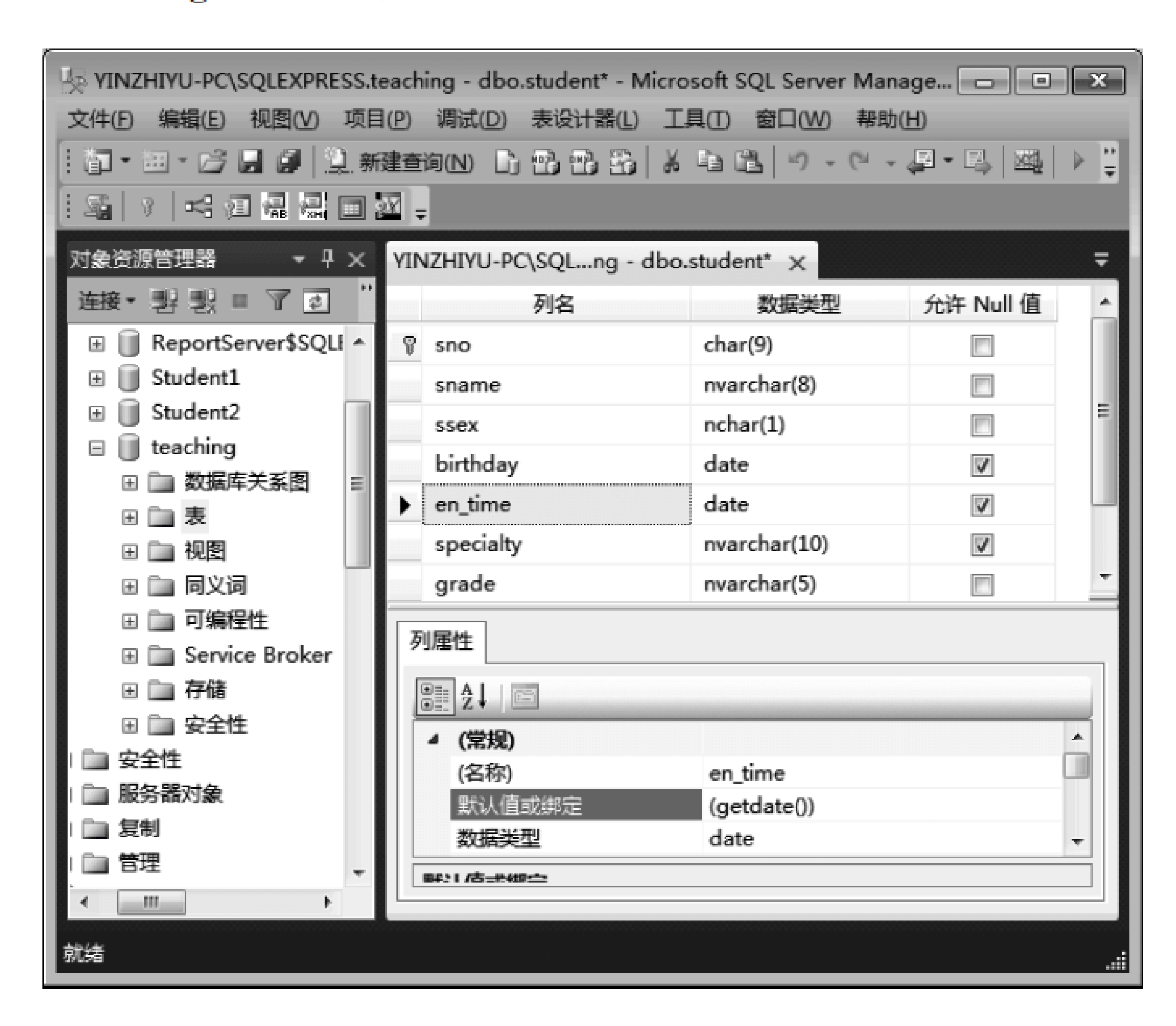


图 3-2　表设计器窗口

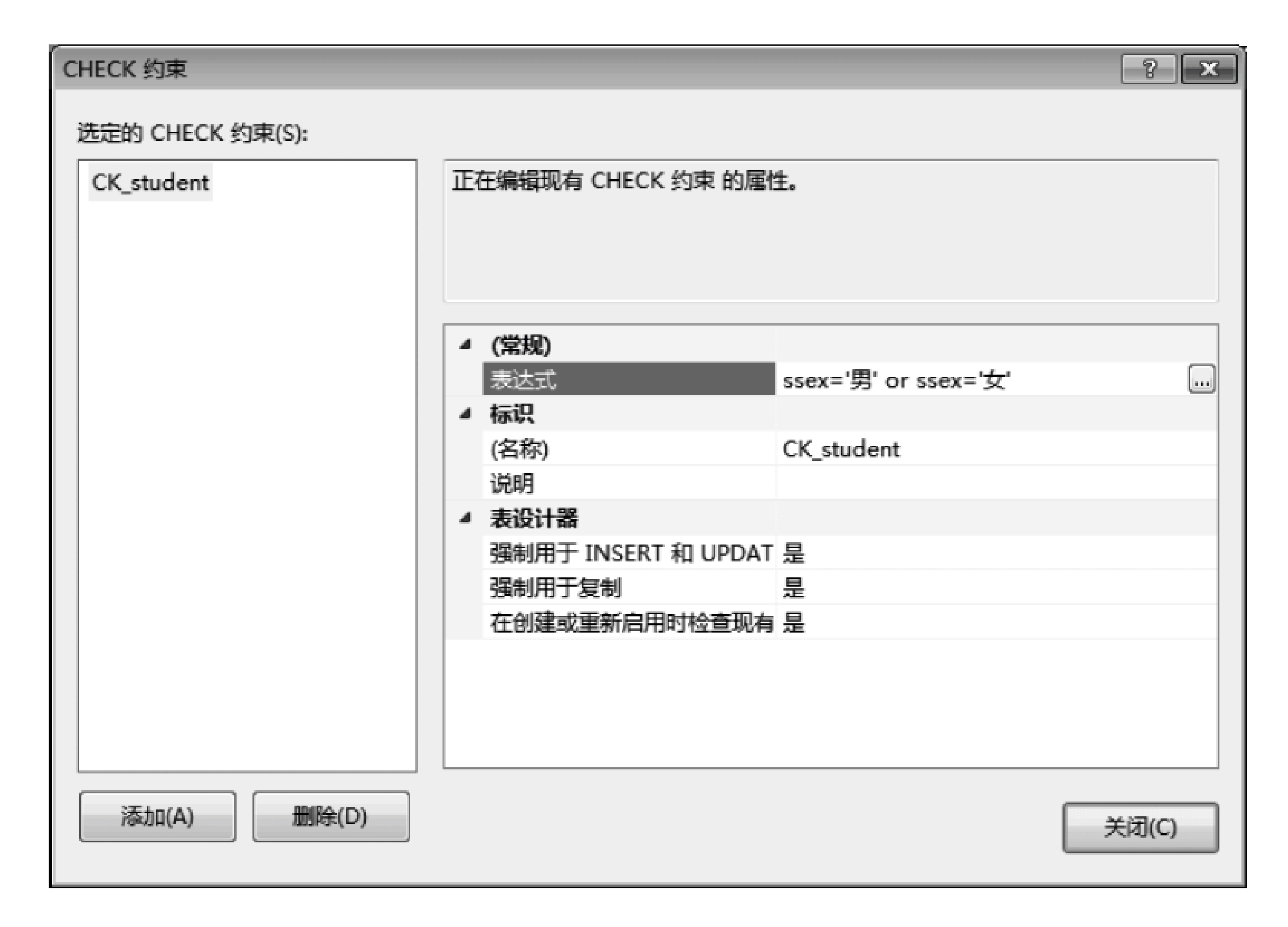


图 3-3　设置性别列的 CHECK 约束

(5) 定义好表中的所有列后,单击保存按钮或“文件”菜单中的“保存表名”命令。在弹出的“选择名称”对话框中,为该表输入一个名称,单击“确定”按钮,如图 3-4 所示。



图 3-4　“选择名称”对话框

3.1.3　使用 T-SQL 语句创建表

在 SQL Server 2008 中可以利用 T-SQL 语句 CREATE TABLE 在数据库中创建表。语法格式如下:

```
CREATE TABLE [ database_name.[ owner ] . | owner. ] table_name
( { < column_definition >
    | column_name AS computed_column_expression
    | < table_constraint >} [, ...n] )
[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]
```

参数说明:

① database_name——用于指定所创建表的数据库名称。database_name 必须是现有数据库的名称。如果不指定数据库,则 database_name 默认为当前数据库。

② owner——用于指定新建表的所有者的用户名,owner 必须是 database_name 所指定的数据库中的现有用户名,owner 默认为当前注册用户名。

③ table_name——用于指定新建表的名称。表名必须符合标识符规则。对于数据库

来说,database_name、owner_name 及 object_name 必须是唯一的。表名最多不能超过 128 个字符。

④ column_name——用于指定新建表的列名。

⑤ computed_column_expression——用于指定计算列的列值表达式。表达式可以是列名、常量、变量、函数等或它们的组合,所谓计算列,是一个虚拟的列,它的值并不实际存储在表中,而是通过对同一个表中其他列进行某种计算而得到的结果。

⑥ ON {filegroup | DEFAULT}——用于指定存储表的文件组名。如果指定 filegroup,则表将存储在指定的文件组中。数据库中必须存在该文件组。如果使用了 DEFAULT 选项,或者省略了 ON 子句,则新建的表会存储在默认的文件组中。

⑦ TEXTIMAGE_ON——用于指定 text、ntext 和 image 列的数据存储的文件组。如果表中没有 text、ntext 或 image 列,则不能使用 TEXTIMAGE_ON。如果没有指定 TEXTIMAGE_ON 子句,则 text、ntext 和 image 列的数据将与数据表存储在相同的文件组中。

上述创建表的语法中< column_definition >包含的内容如下:

```
< column_definition > :: = { column_name data_type }
[ < column_constraint > ] [ , ...n ]
```

其中,< column_constraint >包含的内容如下:

```
< column_constraint > :: = [CONSTRAINT constraint_name]
  {[ NULL | NOT NULL ]
  [ PRIMARY KEY | UNIQUE ]
  [CHECK  ( logical_expression )]
  [DEFAULT {constraint_expression}]
  [FOREIGN KEY [(column )] REFERENCES ref_table [(ref_column)]
  }
```

参数说明:

① NULL 和 NOT NULL——如果表的某一列被指定具有 NULL 属性,那么就允许在插入数据时省略该列的值。反之,如果表的某一列被指定具有 NOT NULL 属性,那么就不允许在没有指定列默认值的情况下插入省略该列值的数据行。在 SQL Server 中列的默认属性是 NULL。

② PRIMARY KEY——设置字段为主码。

③ UNIQUE——设置字段具有唯一性。

④ CHECK——利用逻辑表达式(logical_expression)设置字段的取值范围。

⑤ DEFAULT——利用约束表达式(constraint_expression)设置字段的默认值。

⑥ FOREIGN KEY REFERENCES ref_table [(ref_column)]——设置外键,与其他表建立联系,其中 ref_table 为被参考的主码所在的表名,ref_column 为被参考的主码列名。

注意:使用 T-SQL 语句创建表时,应先打开其所在的数据库。

打开方式:

```
USE 数据库名称
```

【例 3-2】 在数据库 teaching 中创建 course 课程表,表结构如表 3-3 所示。

表 3-3 course 表结构

列名	数据类型	长度	允许空值	键值	取 值 范 围	含义
cno	char	4	否	主码		课程号
cname	nvarchar	20	否			课程名
classhour	tinyint		是		2～6	学时
credit	tinyint		是		2～5	学分

在 SSMS 中新建一个查询窗口,在其中输入如下代码:

```
USE teaching
GO
CREATE TABLE course
 (cno        char(4)      PRIMARY KEY,
  cname      nvarchar(20) NOT NULL ,
  classhour  tinyint CHECK (classhour > = 2 and classhour < = 6),
  credit     tinyint CHECK (credit > = 2 and credit < = 5)
 )
```

执行结果如图 3-5 所示。

图 3-5 使用 T-SQL 语句创建 course 表

3.2 修 改 表

当数据表的结构创建完成后,用户还可以根据实际需要随时更改表结构。用户可以增加、删除和修改字段,更改数据表名称等。

3.2.1 界面方式修改表

（1）在 SSMS 的“对象资源管理器”窗口中，单击“数据库”节点前的“＋”号，展开“数据库”节点；单击目标数据库前的“＋”号，展开目标数据库。

（2）单击“表”节点前的“＋”号，展开“表”节点。在目标表（如 student 表）上右击，弹出快捷菜单，选择“设计”命令，如图 3-6 所示。



图 3-6　修改表菜单

（3）用表设计器向表中添加列，修改列的数据类型、列的数据长度、列的精度、列的小数位数，设置列的为空性等。与创建表时相同。

3.2.2 使用 T-SQL 语句修改表

SQL Server 2008 的 T-SQL 提供的修改表的语句为 ALTER TABLE，其语法格式如下，其参数与创建表的参数含义相同。

```
ALTER  TABLE  table_name
 [ ALTER COLUMN  {column_name                    /* 修改已有列的属性 */
   new_data_type [< column_constraint > ] }
   | ADD{ column_name   data_type                /* 增加新列或约束 */
   [< column_constraint >] | [ CONSTRAINT ] constraint_name }
| DROP                                           /* 删除列或约束 */
{ COLUMN column_name | [ CONSTRAINT ] constraint_name }
```

【例 3-3】 在 student 表中修改 sname 字段的属性，使该字段的数据类型为 nvarchar(10)，允许取空值。

```
USE teaching
```

```
GO
ALTER TABLE student
ALTER COLUMN   sname   nvarchar(10) NULL
```

【例 3-4】 在 course 表中添加 teacher 字段,数据类型为 nvarchar(8)。

```
USE teaching
GO
ALTER TABLE course
ADD teacher nvarchar(8)
```

【例 3-5】 删除 student 表中的 birthday 字段。

```
USE teaching
GO
ALTER TABLE student
DROP COLUMN birthday
```

3.3 列约束和表约束

数据完整性是指关系模型中数据的正确性、有效性和一致性,而约束是通过限制列中数据、行中数据和表之间数据来保证数据完整性的非常有效的方法。其中列约束是针对表中一个列的约束,约束设置在某一个列的列名和数据类型后面;表约束是针对表中一个或多个列的约束,与列定义一样把它定义为表定义的一个表元素。

Microsoft SQL Server 2008 系统提供了 6 种约束类型,即 PRIMARY KEY(主码)、FOREIGN KEY(外键)、UNIQUE(唯一性)、CHECK(取值范围)、DEFAULT(默认值)约束以及是否允许为空(NULL 和 NOT NULL),以下着重介绍前 5 种约束的创建与删除。

3.3.1 PRIMARY KEY 约束

PRIMARY KEY(主码)约束在表中定义一个主码值,在所有的约束类型中,主码约束是最重要的一种约束类型。该约束强制实体完整性,即主码具有唯一性且不允许取空值。

一个表中最多只能有一个主码,主码经常定义在一个列上,但是也可以定义在多个列的组合上,即这些列的组合值不能重复。

1. 创建表时设置主码约束

例 3-2 中创建的 course 表,其主码约束为列约束。

【例 3-6】 在 teaching 数据库中创建名为 sc 的选课表,包括字段 sno、cno、score(成绩),其中 sno、cno 的组合为主码。

```
USE   teaching
GO
CREATE TABLE sc
( sno     char(9),
  cno      char(4),
  score    tinyint,
  CONSTRAINT pk_js PRIMARY  KEY(sno,cno)      /* pk_js 为主码约束名 */
)
```

此例中主码约束为表约束。

创建约束时，可以指定约束的名称。否则，Microsoft SQL Server 系统将提供一个复杂的、系统自动生成的名称。对于一个数据库来说，约束名称必须是唯一的。一般来说，约束的名称应该按照如下格式：

```
约束类型简称_表名_列名_代号
```

2. 用 T-SQL 语句为表添加主码

一般格式为：

```
ALTER TABLE table_name
ADD [ CONSTRAINT constraint_name ]
  PRIMARY KEY
  [CLUSTERED | NONCLUSTERED]                    /* 由系统自动创建聚集或非聚集索引 */
  {( column_name [ ,...n ] ) }
```

【例 3-7】 先在 Student1 数据库中创建表"学生"表，然后通过修改表，对学号字段创建 PRIMARY KEY 约束。

```
USE Student1
GO
CREATE TABLE 学生
(学号      char(6)    NOT NULL,
 姓名       nchar(8)   NOT NULL,
 身份证号   char(18),
 性别       nchar(1)   NOT NULL
)
ALTER TABLE 学生
ADD CONSTRAINT pk_st PRIMARY KEY (学号)
```

3. 删除 PRIMARY KEY 约束

可以使用 ALTER TABLE 的 DROP CONSTRAINT 子句删除 PRIMARY KEY 约束，其一般格式为：

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name [,...n]
```

【例 3-8】 删除 Student1 数据库中"学生"表的 PRIMARY KEY 约束 pk_st。

```
ALTER TABLE 学生
DROP CONSTRAINT pk_st
```

3.3.2 UNIQUE 约束

UNIQUE(唯一性)约束指定表中某一个列或多个列不能有相同的两行或两行以上的数据存在。这种约束通过实现唯一性索引来强制实体完整性。当表中已经有了一个主码约束时，如果需要在其他列上实现实体完整性，又因为表中不能有两个或两个以上的主码约束，所以只能通过创建 UNIQUE 约束来实现。一般地，把 UNIQUE 约束称为候选键约束。

例如，在"学生"表中，主码约束创建在"学号"列上，如果这时还需要保证该表中存储"身

份证号”列的数据是唯一的，那么可以使用 UNIQUE 约束。

1. 创建表时设置 UNIQUE 约束

【例 3-9】 创建“学生 1”表，主码约束创建在“学号”列上，要求“身份证号”列的数据是唯一的。

```
USE Student1
GO
CREATE TABLE 学生 1
( 学号       char(6)   PRIMARY KEY,
  姓名       nchar(8)  NOT NULL,
  身份证号   char(18)  CONSTRAINT uk _st1 UNIQUE,
  性别       bit       NOT NULL
)
```

2. 修改表时设置 UNIQUE 约束

可以使用 ALTER TABLE 的 ADD CONSTRAINT 子句设置 UNIQUE 约束，其一般格式为：

```
ALTER TABLE table_name
  ADD  [ CONSTRAINT constraint_name ]  UNIQUE
  [CLUSTERED | NONCLUSTERED]            /* 由系统自动创建聚集或非聚集索引 */
  ( column_name [ ,...n ] )
```

【例 3-10】 设置“学生”表的“身份证号”字段值唯一。

```
ALTER TABLE 学生
ADD  CONSTRAINT uk_st UNIQUE (身份证号)
```

3. 删除 UNIQUE 约束

方法与删除 PRIMARY KEY 约束相同。

【例 3-11】 删除“学生 1”表中创建的 UNIQUE 约束。

```
ALTER TABLE  学生 1
DROP  CONSTRAINT uk_st1
```

4. 使用 UNIQUE 约束时应考虑的问题

UNIQUE 约束所在的列允许取空值，但是主码约束所在的列不允许取空值；一个表中可以有多个 UNIQUE 约束；可以把 UNIQUE 约束放在一个或者多个列上，这些列或列的组合必须有唯一的值，但是，UNIQUE 约束所在的列并不是表的主码列；UNIQUE 约束强制在指定的列上创建一个唯一性索引，在默认情况下，是创建唯一性的非聚集索引。但是，在定义 UNIQUE 约束时也可以指定所创建的索引是聚集索引。详见第 6 章。

3.3.3 FOREIGN KEY 约束

表和表之间的引用关系可以通过 FOREIGN KEY(外键)约束来实现。创建外键约束既可以由 FOREIGN KEY 子句完成，也可以在表设计器中完成。

1. 在图形界面下建立表之间的关系

在 SSMS 的对象资源管理器中创建表之间的关系图，可以实现表连接，即外键约束。

步骤如下：

(1) 在“对象资源管理器”窗口中，展开数据库，如 teaching，选择“数据库关系图”，右击，如图 3-7 所示。



图 3-7　新建数据库关系图菜单

(2) 在弹出的快捷菜单中选择“新建数据库关系图”命令，出现“添加表”对话框，如图 3-8 所示。



图 3-8　“添加表”对话框

(3) 选择要建立关联的表，单击“添加”按钮添加表，关闭“添加表”对话框，用鼠标左键拖动不同表上相关的属性前的按钮，如 student 表的 sno 列和 sc 表的 sno 列，出现表的关联关系，如图 3-9 所示。

(4) 关联图建好后，在关闭时会弹出提示是否保存更改对话框，如图 3-10 所示。

(5) 若要保存更改，则单击“是”按钮，出现“选择名称”对话框，如图 3-11 所示，输入关系图名称，单击“确定”按钮。表间关系创建完毕。

外键约束定义一个或多个列，这些列可以引用同一个表或另外一个表中的主码约束列或 UNIQUE 约束列。实际上，通过创建外键约束可以实现表和表之间的依赖关系。

一般情况下，在 Microsoft SQL Server 关系型数据库管理系统中，表和表之间经常存在着大量的关系，这些关系都是通过定义主码约束和外键约束实现的。

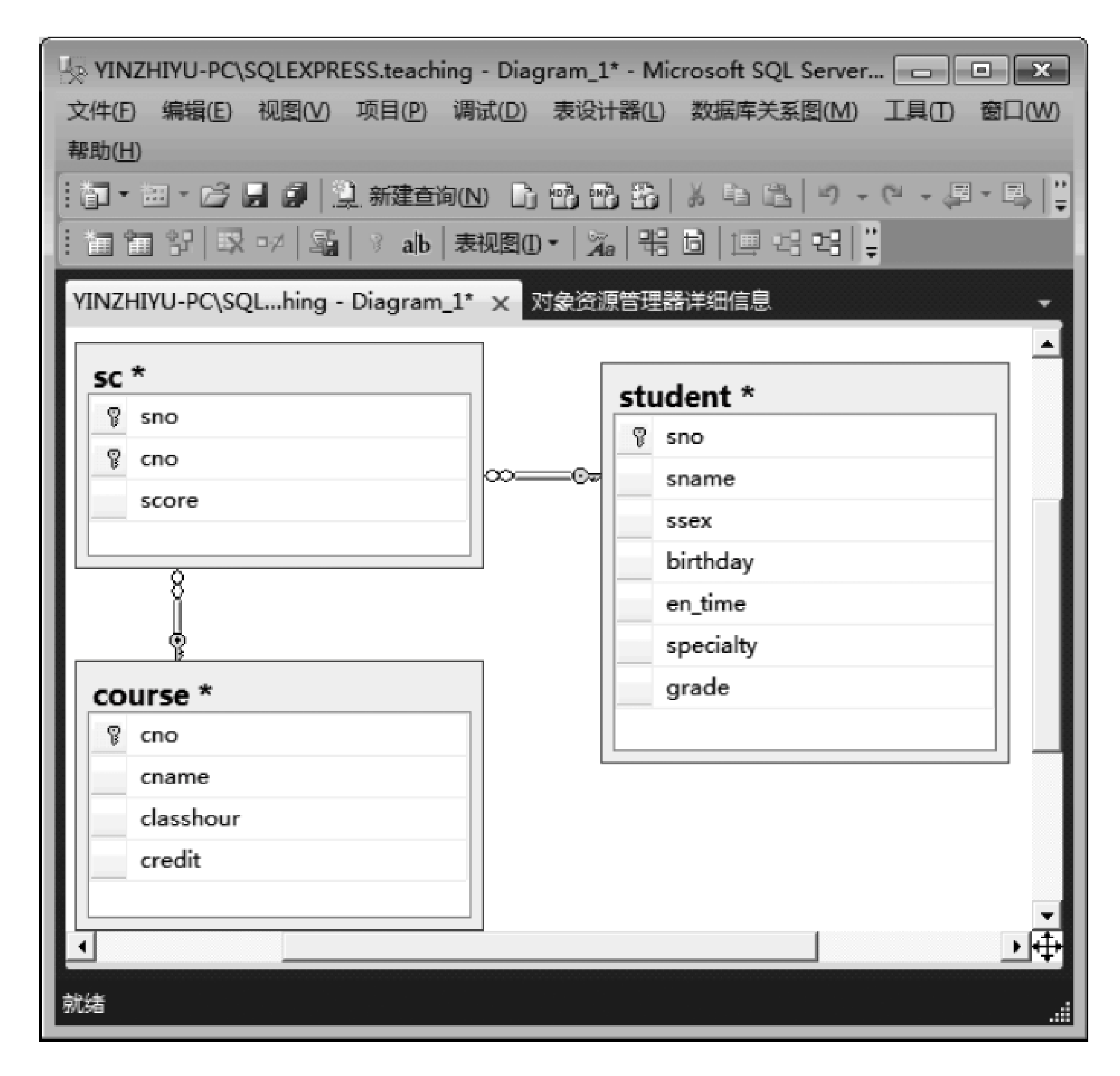


图 3-9 表的关联图



图 3-10 提示是否保存更改对话框



图 3-11 "选择名称"对话框

2. 创建表时定义外键约束

【例 3-12】 在数据库 Student1 中创建一个“成绩”表，包括学号、课程号、成绩，并为成绩表创建外键约束，该约束把“成绩”表中的“学号”字段和“学生”表中的“学号”字段关联起来。

```
USE Student1
GO
CREATE TABLE 成绩
( 学号   char(6) CONSTRAINT  st_xh
  FOREIGN  KEY  REFERENCES 学生(学号),
  课程号 char(4),
  成绩   int )
```

3. 修改表时添加外键约束

语法格式如下：

```
ALTER TABLE table_name
ADD [ CONSTRAINT constraint_name]
FOREIGN KEY { ( column_name [ ,…n ] )}
REFERENCES {ref_table ( ref_column [ ,…n ] ) }
```

【例 3-13】 将 teaching 数据库中 student 表、course 表和 sc 表进行关联，student 表和 course 表为主表，其中的 sno 和 cno 字段为主码；sc 表为从表，将 sc 表的 sno 和 cno 字段定义为外键。

```
① USE teaching
   GO
   ALTER TABLE sc
   ADD CONSTRAINT st_foreign
       FOREIGN KEY   (sno)  REFERENCES  student(sno)
② USE teaching
   GO
   ALTER TABLE sc
   ADD CONSTRAINT kc_foreign
       FOREIGN KEY   (cno)  REFERENCES  course(cno)
```

4. 删除外键约束

T-SQL 语句的 ALTER TABLE 命令可以删除外键约束。

【例 3-14】 删除例 3-12 创建的外键约束。

```
USE Student1
GO
ALTER TABLE 成绩
DROP  CONSTRAINT  st_xh
```

5. 级联更新和级联删除

外键是双向的，无论用户在参照表做了什么，外键都将检查被参照表，以保持外键和主码的一致性，避免出现不完整的记录。

对于 SQL Server 而言，默认情况下如果被参照表中的某行数据的主码被引用，那么将不允许对该行删除或修改其主码值。但是若希望在删除被参照表数据或修改被参照表中某

个主码值的同时，自动删除参照表中对应的行，或将对应行的外键列同时修改，或将对应行的外键列设置为 NULL 等等，那么将用到级联更新和删除。

级联更新和删除是 FOREIGN KEY 约束语法中的一部分，其语法格式如下：

(1) 级联删除。

```
[ON DELETE {NO ACTION|CASCADE|SET NULL|SET DEFAULT }]
```

(2) 级联更新。

```
[ON UPDATE {NO ACTION|CASCADE|SET NULL|SET DEFAULT }]
```

其中，ON DELETE 表示级联删除操作，ON UPDATE 表示级联更新操作。

NO ACTION 是 SQL Server 的默认选项，表示不允许对被参照表执行删除或更新操作。CASCADE 是层叠操作，表示级联自动删除或更新参照表相关数据。SET NULL 表示将参照表中的外键列数据设置为 NULL，如果外键列定义了 NOT NULL 约束，则不能使用该选项。SET DEFAULT 表示将参照表中的外键列数据设置为默认值，如果外键列未定义 DEFAULT 值，则不能使用该选项。

【例 3-15】 创建一个学生表 stud(学号，姓名，性别)，主码为学号；创建一个选课表 sc(学号，课程号，成绩)，其中学号为外键，并设置级联删除和级联更新。

```
CREATE TABLE stud
 ( 学号  char(6) PRIMARY KEY,
   姓名 nvarchar(8),
   成绩   int )
CREATE TABLE sc
( 学号  char(6) FOREIGN KEY REFERENCES stud(学号)
  ON DELETE CASCADE ON UPDATE CASCADE,
  课程号 char(4),
  成绩   int )
```

也可以通过关系图设置级联删除和级联更新。例如设置 teaching 数据库中 student 表和 sc 表的级联更新和级联删除，操作步骤如下：

(1) 右击 teaching 数据库中的关系图，选择“修改”命令，如图 3-12 所示。



图 3-12　修改关系图菜单

(2) 进入数据库关系图界面，右击 student 表和 sc 表之间的关系图，从弹出的快捷菜单中选择“属性”命令，如图 3-13 所示。

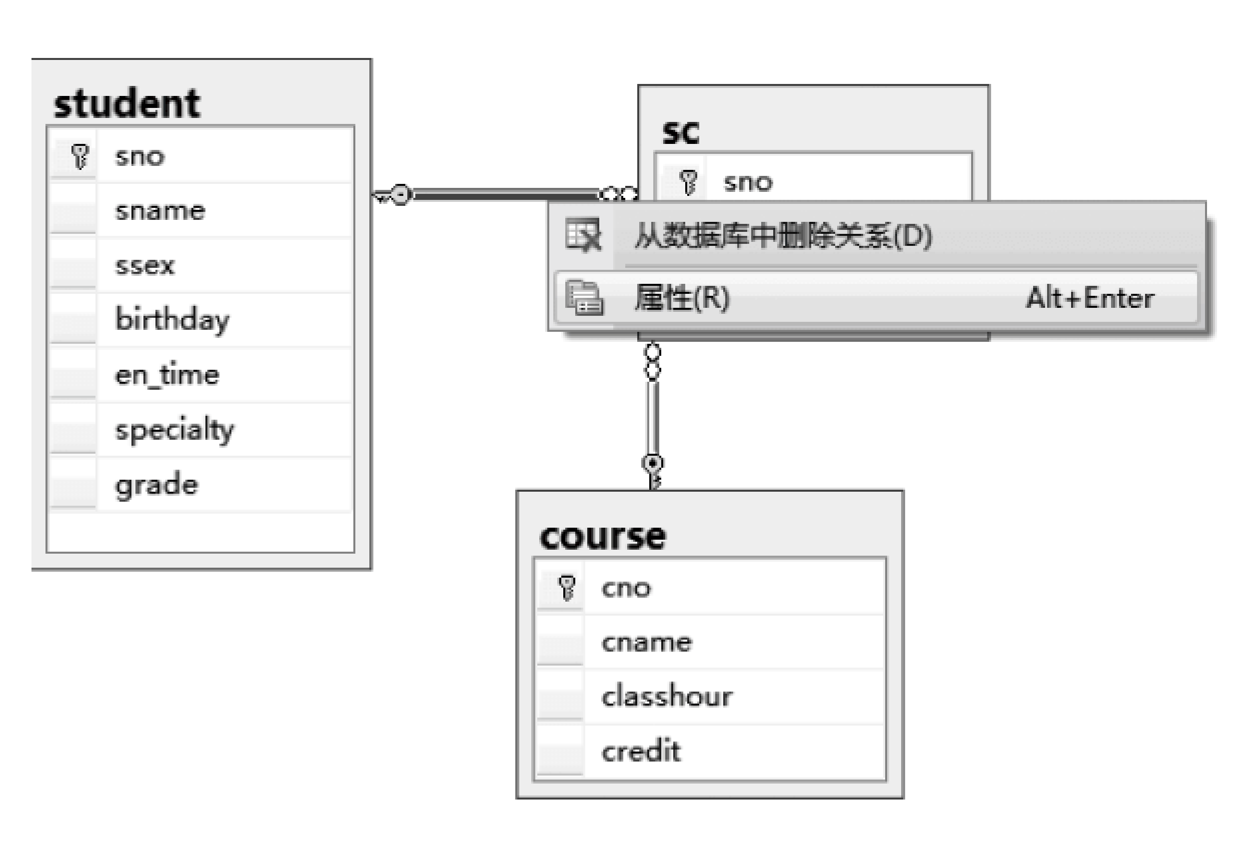


图 3-13　数据库关系图界面

(3) 在“属性”窗口中，展开“INSERT 和 UPDATE 规范”选项，在“更新规则”和“删除规则”右侧分别选择“级联”选项，如图 3-14 所示。

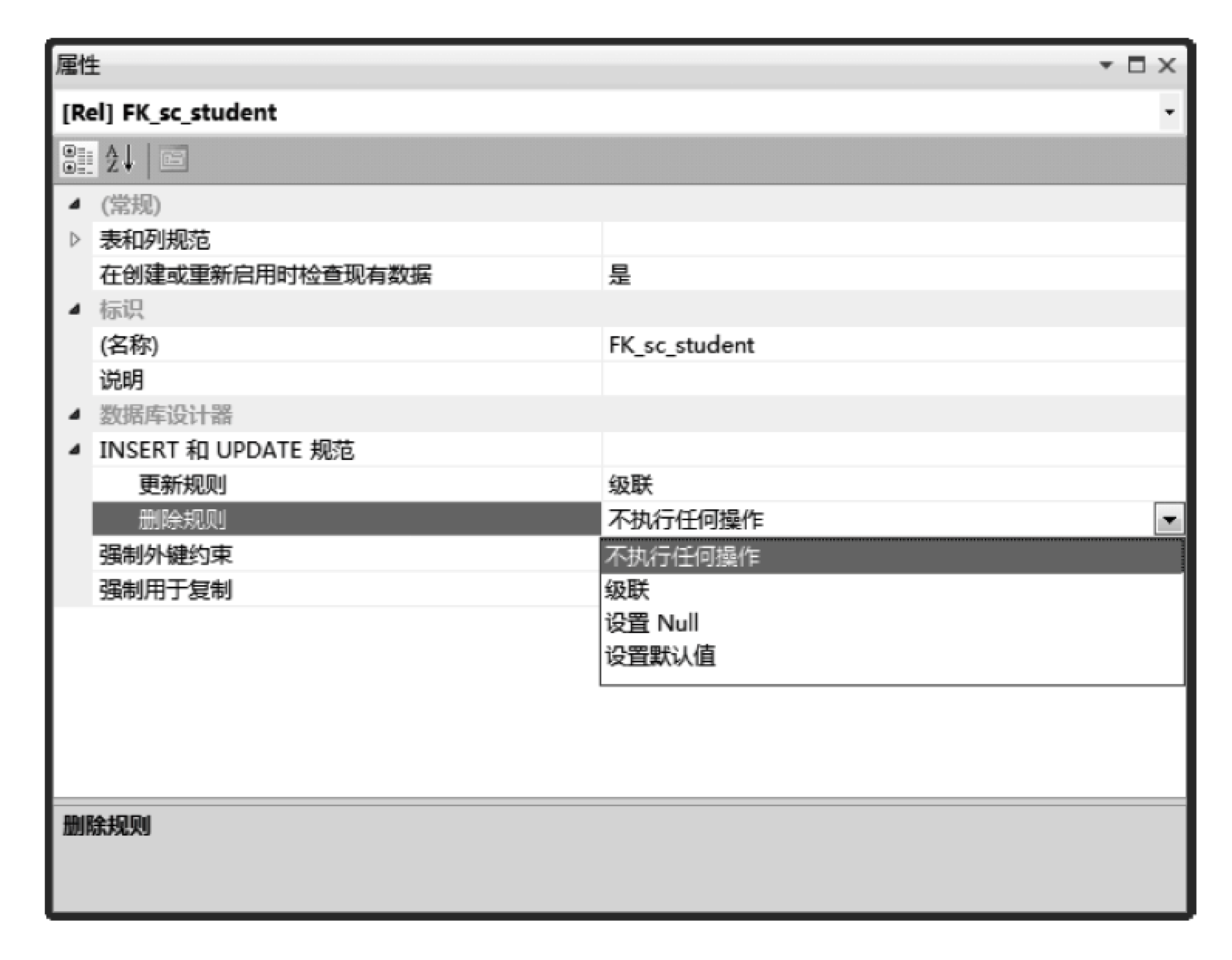


图 3-14　设置级联更新和级联删除窗口

(4) 关闭“属性”窗口，回到数据库关系图界面，保存关系图即可。

下面可尝试修改某个学生的学号或删除某个学生的信息，体验级联更新和级联删除的应用。

注意：由于数据库操作人员对 UPDATE 和 DELETE 命令执行的操作关注度较差，所以级联更新和级联删除要谨慎使用，特别是对于比较复杂或级联层次较多的数据库，级联后影响的深度可能是无限的。

3.3.4　CHECK 约束

CHECK 约束又叫域完整性约束，用来限制用户输入某一个列的数据，即在该列中只能

输入指定范围的数据。CHECK 约束的作用非常类似于外键约束，两者都是限制某个列的取值范围，但是外键是通过其他表来限制列的取值范围，CHECK 约束是通过指定的逻辑表达式来限制列的取值范围。

例如，在描述学生“性别”列中可以创建一个 CHECK 约束，指定其取值范围是“男”或者“女”。这样，当向该列输入数据时，要么输入数据“男”，要么输入数据“女”，而不能输入其他不相关的数据。

1. 创建表时创建 CHECK 约束

T-SQL 语句在创建表时创建 CHECK 约束的语法格式如下：

```
CREATE TABLE table_name                          /* 指定表名 */
(column_name  datatype
[check_name ] CHECK  ( logical_expression )      /* CHECK 约束表达式 */
    [,…n])
```

【例 3-16】 在 Student1 数据库中创建表 books，其中包含 CHECK 约束（列约束）。

```
USE Student1
GO
CREATE TABLE books
(
   book_id      smallint        PRIMARY KEY,        /* 书号 */
   book_name    nvarchar(20)    NOT NULL,           /* 书名 */
   max_lvl      tinyint         NOT NULL   CHECK   (max_lvl <= 250)
                                                    /* 书允许的最高价 CHECK 约束 */
)
```

【例 3-17】 在 Student1 数据库中创建表——身份信息，其中包含 CHECK 约束（表约束）。

```
USE Student1
GO
CREATE TABLE 身份信息
(
    id_no      char(18)        PRIMARY KEY,       /* 身份证号 */
    name       nvarchar(10)    NOT NULL,          /* 姓名 */
    startdate  date            NOT NULL,          /* 有效开始日期 */
    enddate    date            NOT NULL,          /* 有效结束日期 */
    CHECK (startdate < enddate)
)
```

2. 修改表时创建 CHECK 约束

语法格式如下：

```
ALTER TABLE table_name
ADD CONSTRAINT check_name   CHECK   (logical_expression)
```

【例 3-18】 通过修改 Student1 数据库的“成绩”表，增加成绩字段的 CHECK 约束。

```
USE Student1
GO
```

```
ALTER TABLE 成绩
ADD CONSTRAINT cj_constraint  CHECK  (成绩>=0 and 成绩<=100)
```

3. 删除 CHECK 约束

T-SQL 语句的 ALTER TABLE 命令可以删除 CHECK 约束。

【例 3-19】 删除例 3-17 创建的 CHECK 约束。

```
USE Student1
GO
ALTER TABLE 成绩
DROP CONSTRAINT cj_constraint
```

一个列上可以定义多个 CHECK 约束；当执行 INSERT 语句或者 UPDATE 语句时，该约束验证相应的数据是否满足 CHECK 约束的条件。但是，执行 DELETE 语句时不检查 CHECK 约束。

3.3.5 DEFAULT 约束

当使用 INSERT 语句插入数据时，如果没有为某一个列指定数据，那么 DEFAULT(默认值)约束就在该列中输入一个默认值。

例如，在学生表的“性别”列中定义了一个 DEFAULT 约束为“男”。当向该表中输入数据时，如果没有为“性别”列提供数据，那么 DEFAULT 约束把默认值“男”自动插入到该列中。因此，DEFAULT 约束可以实现保证域完整性。

1. 创建表时定义 DEFAULT 约束

语法格式如下：

```
CREATE TABLE table_name                          /*指定表名*/
(column_name  datatype
DEFAULT constraint_expression                    /*默认值约束表达式*/
 [,...n])
```

【例 3-20】 先在 Student1 数据库中创建学生表 st，定义一个“入学日期”字段的默认值为系统当前日期。

```
USE Student1
GO
CREATE TABLE st
   (  学号      char(6)         PRIMARY KEY,
      姓名      nchar(8)        NOT NULL,
      专业名    nvarchar(20)    NULL,
      性别      bit             NOT NULL,
      出生日期  date            NOT NULL,
      备注      text            NULL,
      入学日期  date            DEFAULT  getdate()  /*定义默认值约束*/
   )
```

说明：因为没有使用“CONSTRAINT 约束名”，所以使用系统定义的名称。

2. 修改表时定义 DEFAULT 约束

【例 3-21】 修改 st 表，添加一字段“总学分”，并为其设置默认值约束，默认值为 150。

```
USE Student1
GO
ALTER TABLE ST
ADD  总学分  smallint  NULL
    CONSTRAINT  totaldf                    /* 默认值约束名 */
    DEFAULT  150
```

3. 删除 DEFAULT 约束

【例 3-22】 删除例 3-20 定义的默认值约束。

```
USE Student1
GO
ALTER TABLE st
DROP CONSTRAINT totaldf
```

4. 定义 DEFAULT 约束时应考虑的问题

① 定义的常量值与该列的数据类型和精度必须是一致的。

② DEFAULT 约束只能应用于 INSERT 语句。

③ 每一个列只能定义一个 DEFAULT 约束。DEFAULT 约束不能放在有 IDENTITY 属性的列上或者数据类型为 timestamp 的列上，因为这两种列都会由系统自动提供数据。

④ DEFAULT 约束允许指定一些由系统函数提供的值，这些系统函数包括 SYSTEM_USER、GETDATE、CURRENT_USER 等。

注意：[NOT] NULL 和 DEFAULT 约束只能设置一个列的约束，所以只能使用列约束；PRIMARY KEY、UNIQUE、CHECK 和 FOREIGN KEY 约束，一个列的约束既可以使用列约束，也可以使用表约束，多个列的约束只能使用表约束。

3.4 表数据操作

表的基本结构建好后，表内没有数据，可以在 SSMS 中利用图形界面非常方便地对数据执行各种操作，也可以利用 T-SQL 中的 INSERT 命令完成相应的功能。

3.4.1 向表中添加数据

1. 界面方式输入数据

(1) 在“对象资源管理器”窗口中，展开数据库和表，右击表名，如 teaching 中的 student 表，弹出快捷菜单，选择“编辑前 200 行”命令，如图 3-15 所示。

(2) 出现一个空表，如图 3-16 所示。

(3) 输入数据，如图 3-17 所示，需注意以下几点：

① 注意约束。PRIMARY KEY(具有唯一性、不允许为空)和 NOT NULL(不允许为空)。



图 3-15　选择"编辑前 200 行"命令

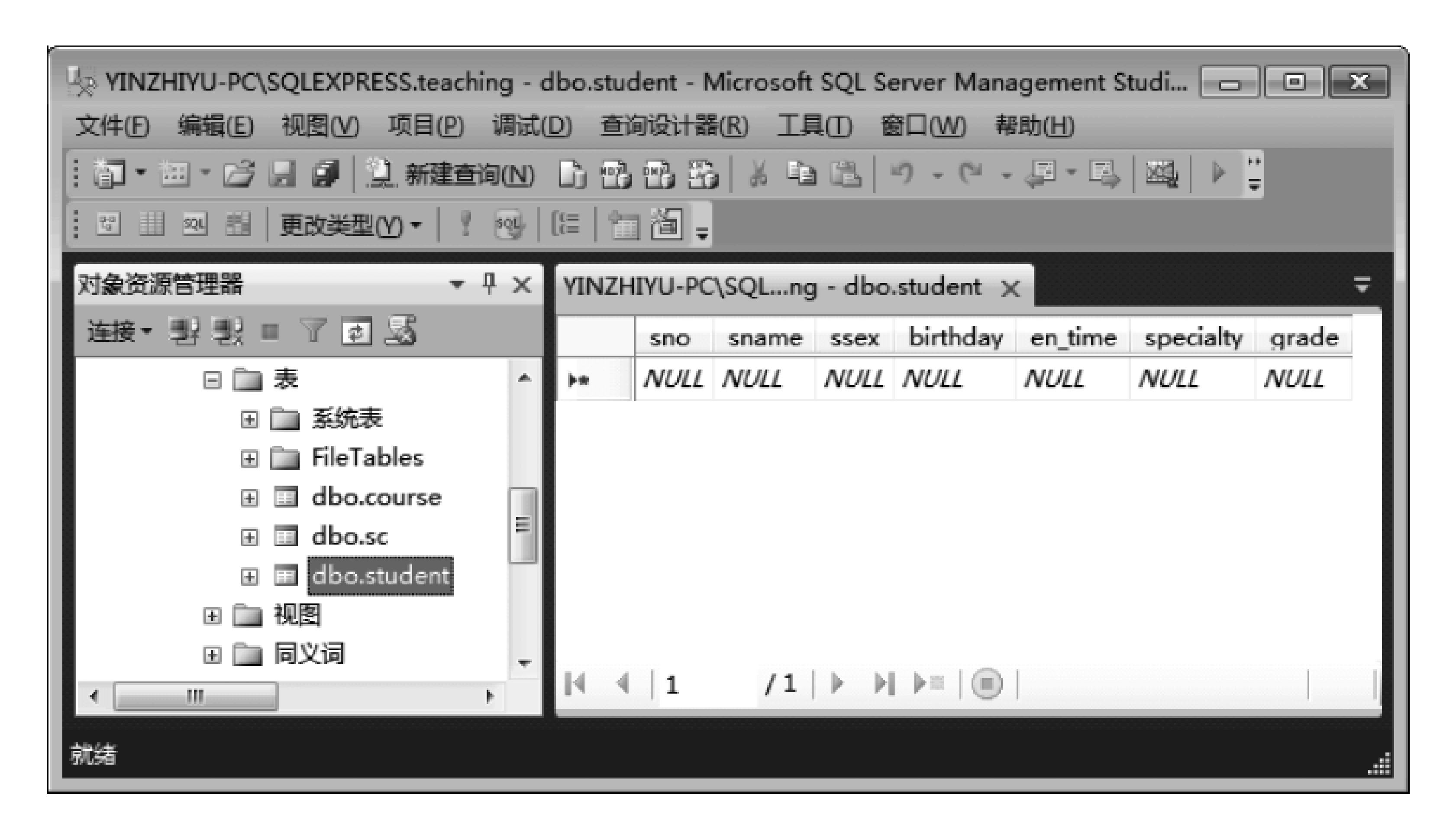
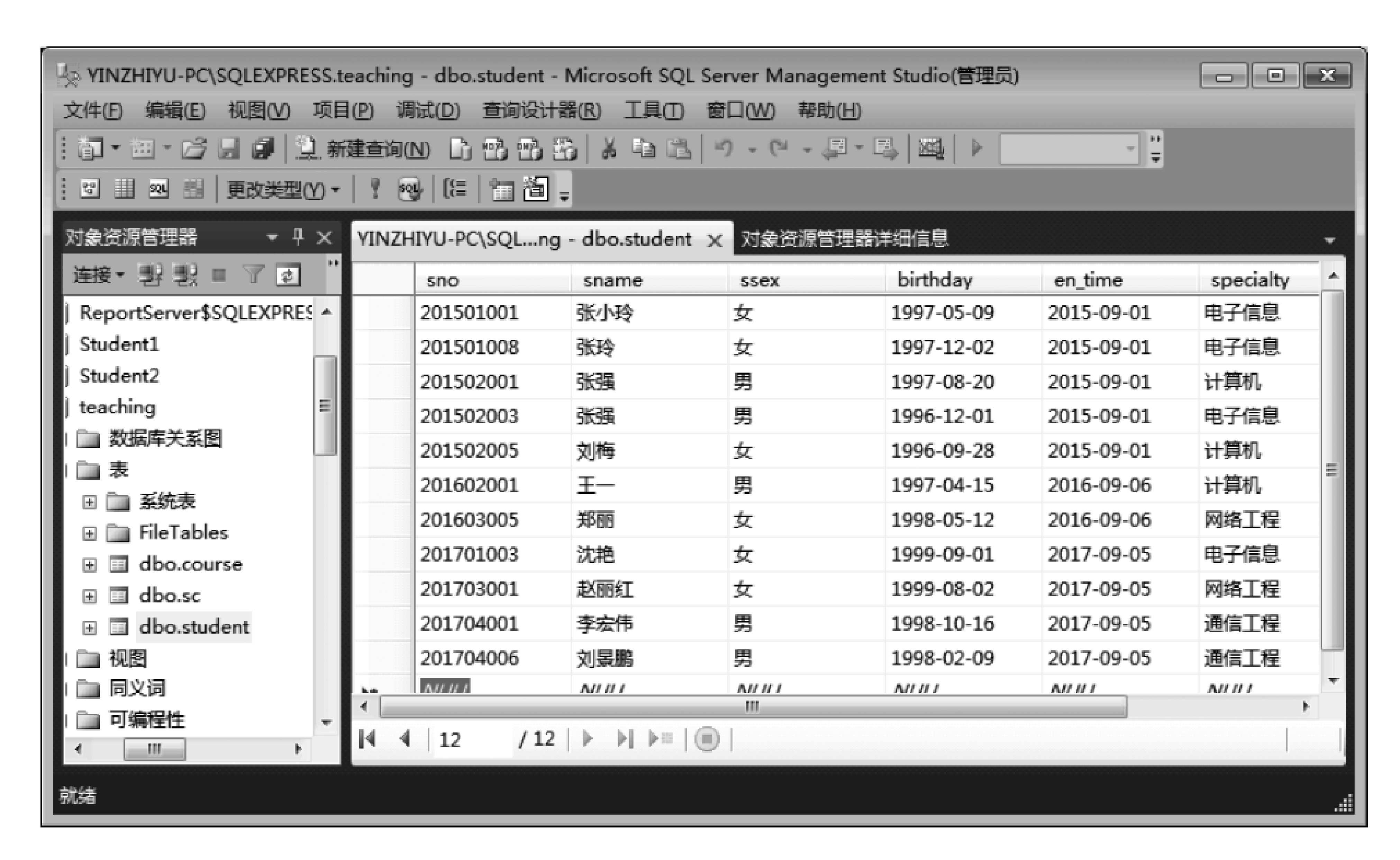


图 3-16　空表窗口



sno	sname	ssex	birthday	en_time	specialty
201501001	张小玲	女	1997-05-09	2015-09-01	电子信息
201501008	张玲	女	1997-12-02	2015-09-01	电子信息
201502001	张强	男	1997-08-20	2015-09-01	计算机
201502003	张强	男	1996-12-01	2015-09-01	电子信息
201502005	刘梅	女	1996-09-28	2015-09-01	计算机
201602001	王一	男	1997-04-15	2016-09-06	计算机
201603005	郑丽	女	1998-05-12	2016-09-06	网络工程
201701003	沈艳	女	1999-09-01	2017-09-05	电子信息
201703001	赵丽红	女	1999-08-02	2017-09-05	网络工程
201704001	李宏伟	男	1998-10-16	2017-09-05	通信工程
201704006	刘昱鹏	男	1998-02-09	2017-09-05	通信工程

图 3-17　数据表窗口

② 如果数据行前出现红色叹号，表示数据可能有问题；如果确实有问题而修改后叹号还不消失，可以按 Esc 键重新录入本行数据。

③ 录入完一行数据无须保存，按回车键或录入下一行时上一行已自动保存。

2. 用 INSERT 命令插入数据

T-SQL 中提供 INSERT 命令向表中插入数据，其语法格式如下：

```
INSERT [ INTO] table_name  [ ( column_name_list ) ]
{ VALUES( expression [ ,…n] ) }
```

其中参数的含义如下：

① INTO——一个可选的关键字，可以将它用在 INSERT 和目标表之间。

② table_name——将要接收数据的表或 table 变量的名称。

③ column_list——要在其中插入数据的一列或多列的列表。必须用小括号将 column_list 括起来，并且用逗号进行分隔。

④ VALUES——引入要插入的数据值的列表。对于 column_list 中或者表中的每个列，都必须有一个数据值。必须用小括号将值列表括起来。如果 VALUES 列表中的值与表中列的顺序不相同，或者未包含表中所有列的值，那么必须使用 column_list 明确地指定存储每个传入值的列。

⑤ expression——列值表达式。

【例 3-23】 在 teaching 数据库的 student 表中插入一行数据(sno、sname、ssex、grade)为('201601015','刘玲玲','女','2016 级')。

```
USE teaching
GO
INSERT into student(sno,sname,ssex,grade)
VALUES('201601015','刘玲玲','女','2016 级')
```

【例 3-24】 在 Student1 数据库的“学生”表中插入一行数据('160101','刘玲','130212199807190926','女')。

```
USE Student1
GO
INSERT into 学生 VALUES('160101','刘玲','130212199807190926','女')
```

【例 3-25】 在 Student1 数据库的“学生”表中同时插入三行数据，插入命令如下：

```
INSERT into 学生 VALUES('160102','王小玲','130212199707190926','女'),('160103','王伟',
'130212199809100871','男'), ('160104','张大力','130212199802150812','男')
```

执行结果如图 3-18 所示。

注意： 创建表时设置为不允许为空、又没有默认值约束的列，必须插入数据。

3.4.2 修改表中数据

1. 界面方式修改数据

在 SSMS 中，选择相应的表，右击后，在弹出的快捷菜单中选择“编辑前 200 行”命令，出现编辑表数据窗口，在该窗口中，可以直接对数据进行修改操作。

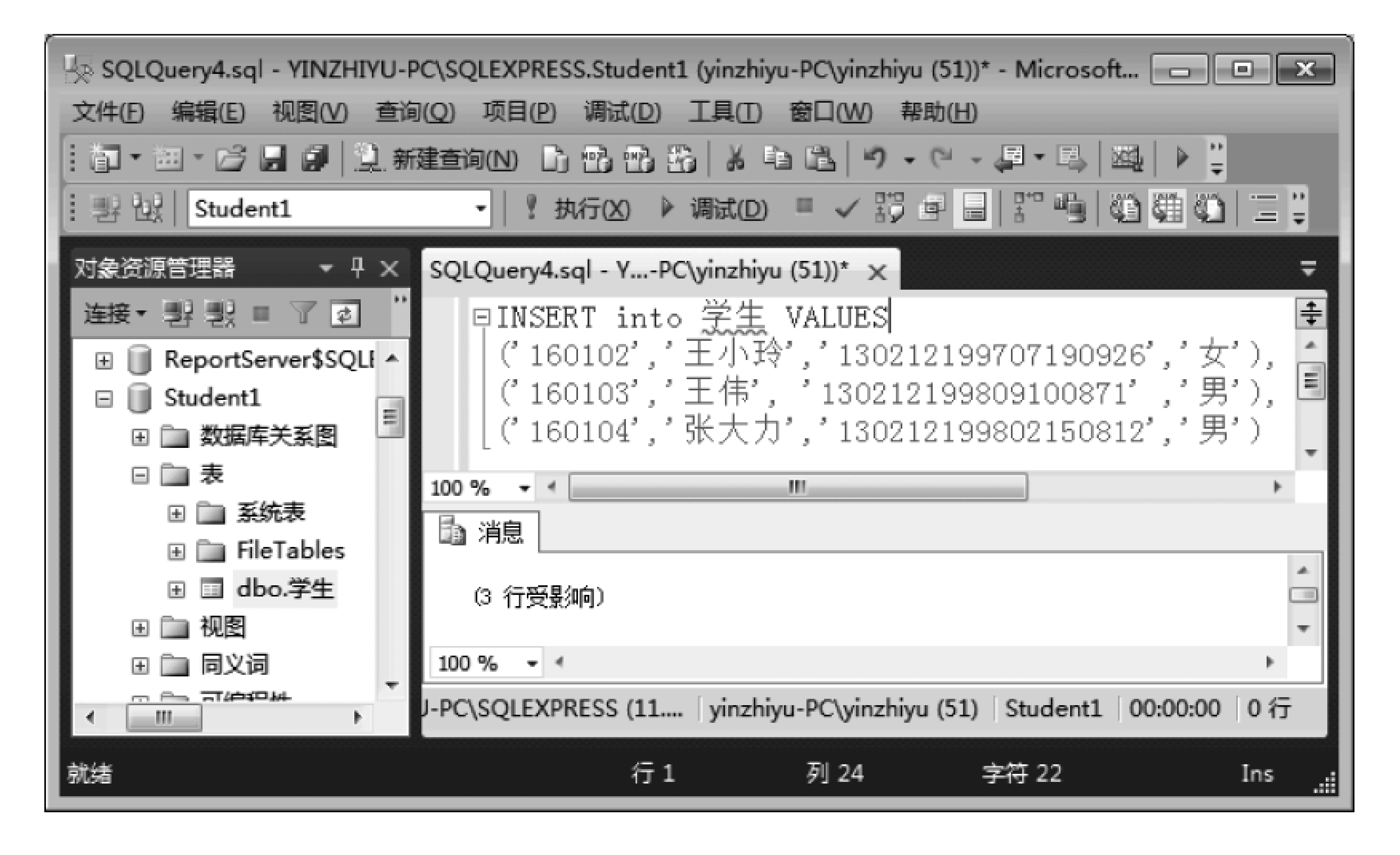


图 3-18 向表中插入多行数据

2. 用 UPDATE 语句修改数据

T-SQL 中提供 UPDATE 命令修改表中数据，其语法格式如下：

```
UPDATE table_name
SET { column_name = expression } [ ,...n ]
  [WHERE {condition_expression}]
```

其中，参数的含义如下：

① table_name——需要更新的表的名称。

② SET——指定要更新的列或变量名称的列表。

③ column_name——含有要更改数据的列的名称。

④ expression——列值表达式。

⑤ condition_expression——条件表达式。对条件的个数没有限制。

⑥ 如果没有 WHERE 子句，则 UPDATE 将会修改表中的每一行数据。

【例 3-26】 将 Student1 数据库的“学生”表中所有学生的“性别”值都改为“男”。

```
USE Student1
GO
UPDATE 学生
SET 性别 = '男'
```

【例 3-27】 在 Student1 数据库的“学生”表中添加一字段“备注”nvarchar(20)，“备注”字段信息为“已毕业”。

①
```
USE Student1
GO
ALTER TABLE 学生
Add  备注 nvarchar(20)
```

②
```
UPDATE 学生
SET 备注 = '已毕业'
```

注意： 新添加字段一定要设置“允许为空”，否则会添加失败。

【例 3-28】 将 Student1 数据库的“学生”表中学号为 160101 的学生姓名改为“王武”。

```
USE Student1
GO
UPDATE 学生 SET 姓名 = '王武'
WHERE 学号 = '160101'
```

3.4.3 删除表中数据

1. 界面方式删除数据

在 SSMS 中，选择相应的表，右击后，在弹出的快捷菜单中选择“编辑前 200 行”命令，出现编辑表数据窗口，在该窗口中选择要删除的记录，右击后，在弹出的快捷菜单中选择“删除”命令。

2. 用 DELETE 语句删除数据

T-SQL 中提供 DELETE 命令删除表中数据，其语法格式如下：

```
DELETE [FROM] table_name
[ WHERE {condition_expression} ]
```

其中，参数的含义如下：

① table_name——是要从其中删除行的表的名称。

② WHERE——指定用于限制删除行数的条件。如果没有提供 WHERE 子句，则删除表中的所有行。

③ condition_expression——指定删除行的限定条件。对条件的个数没有限制。

【例 3-29】 删除 Student1 数据库的“学生”表中 160101 号学生的记录。

```
USE Student1
DELETE  学生 WHERE 学号  =  '160101'
```

3. 用 TRUNCATE TABLE 清空表格

语法格式如下：

```
TRUNCATE  TABLE  table_name
```

其中，table_name 为要删除所有记录的表名。

TRUNCATE TABLE 与不含有 WHERE 子句的 DELETE 语句在功能上相同。但是，TRUNCATE TABLE 速度更快，并且使用更少的系统资源和事务日志资源。

【例 3-30】 清空“学生”表中的数据。

```
TRUNCATE TABLE 学生
```

3.5 删 除 表

删除表就是将表中数据和表的结构从数据库中永久性地去除。表被删除之后，就不能再恢复该表的定义。

1. 界面方式删除表

打开 SSMS,展开“数据库”,再展开“表”,右击要删除的表,在弹出的快捷菜单中选择“删除”命令,如图 3-19 所示,在随后出现的“删除对象”对话框中单击“确定”按钮,即可完成指定表的删除操作。



图 3-19 “删除”命令

2. 用 T-SQL 语句删除表

可以利用 T-SQL 语句中的 DROP TABLE 命令删除一个或多个数据表。

其语法格式如下:

```
DROP TABLE table_name[,...n]
```

【例 3-31】 删除 student1 数据库中“学生”表。

```
DROP TABLE 学生
```

3.6 数据的导入/导出

通过导入和导出数据的操作可以在 SQL Server 2008 和其他异类数据源(例如 Excel 或 Oracle 数据库)之间轻松移动数据。“导出”是指将数据从 SQL Server 表复制到数据文件。“导入”是指将数据从数据文件加载到 SQL Server 表。

在 SQL Server 2008 中导入和导出数据的操作可以在 SSMS 中使用向导来完成,也可以通过执行 T-SQL 语句来完成。本书只介绍前一种方法的数据库数据的导入/导出。

3.6.1 导出数据

数据的导出是将一个 SQL Server 数据库中的数据导出到一个文本文件、电子表格或其他格式的数据库中。

【例 3-32】 将 teaching 中 student 表导出至“G:\data”文件夹下形成 st. xls 文件。

具体操作步骤如下：

① 在“对象资源管理器”中，展开数据库，右击要导出数据所在的数据库 teaching，弹出快捷菜单，选择“任务”级联菜单中的“导出数据”命令，出现“SQL Server 导入和导出向导”中的“欢迎使用 SQL Server 导入和导出向导”窗口，如图 3-20 和图 3-21 所示。

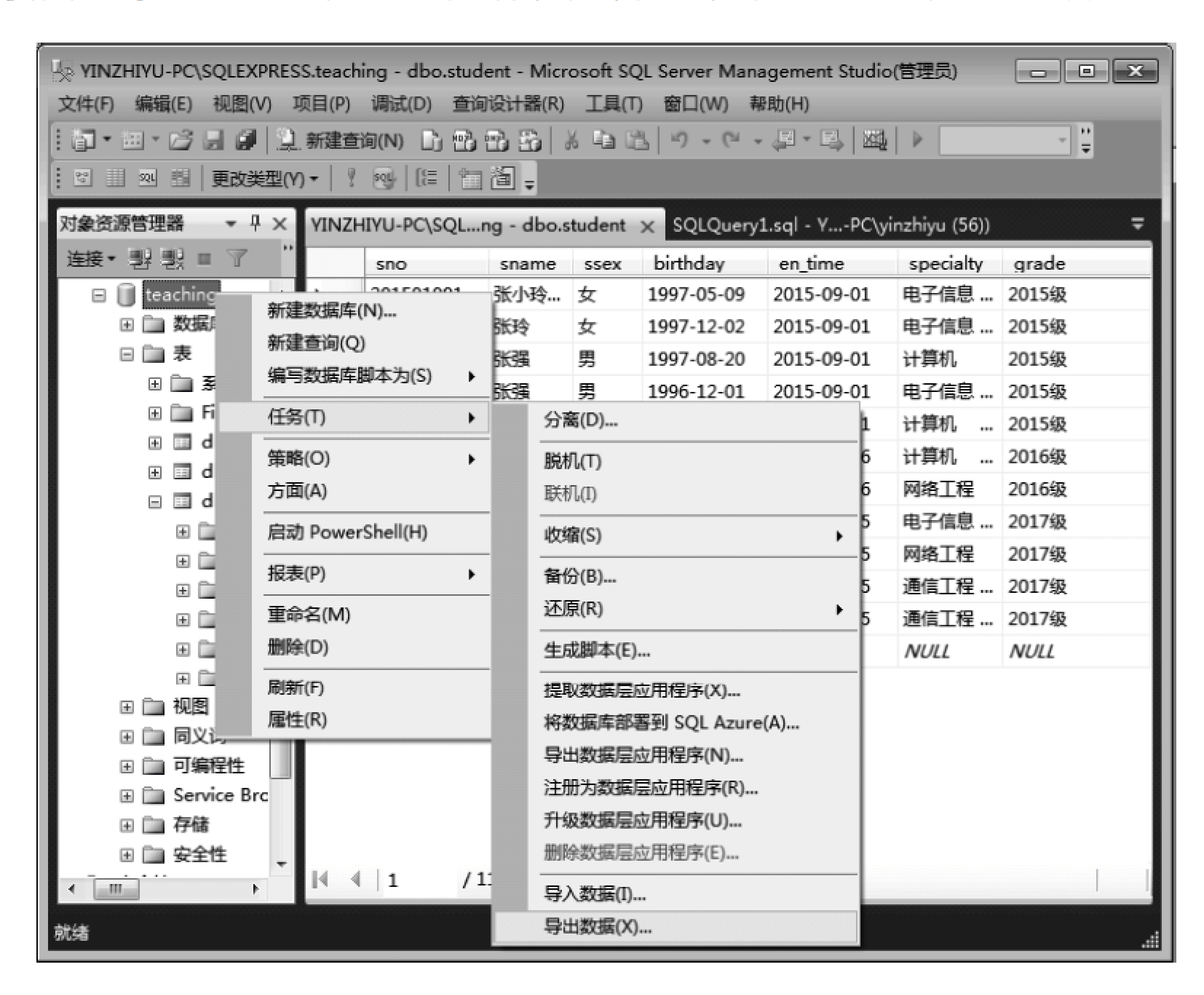


图 3-20 “导出数据”命令



图 3-21 导入和导出向导

② 单击“下一步”按钮，出现“选择数据源”窗口，如图 3-22 所示。



图 3-22 “选择数据源”窗口

③ 单击“下一步”按钮，出现“选择目标”窗口，如图 3-23 所示。



图 3-23 “选择目标”窗口

④ 单击“下一步”按钮，出现“指定表复制或查询”窗口，可以选择复制整表还是表中的部分数据，如果复制整表数据，选择第一个单选按钮；如果导出表中的部分数据，则选中第二个单选按钮，并需通过编写 SQL 查询语句来实现，如图 3-24 所示。



图 3-24 “指定表复制或查询”窗口

⑤ 这里选中第一个单选按钮，然后单击“下一步”按钮，出现“选择源表和源视图”窗口，如图 3-25 所示。



图 3-25 “选择源表和源视图”窗口

⑥ 单击“编辑映射”按钮，出现“列映射”窗口，如图 3-26 所示，可以在此对“列属性”等内容进行修改。

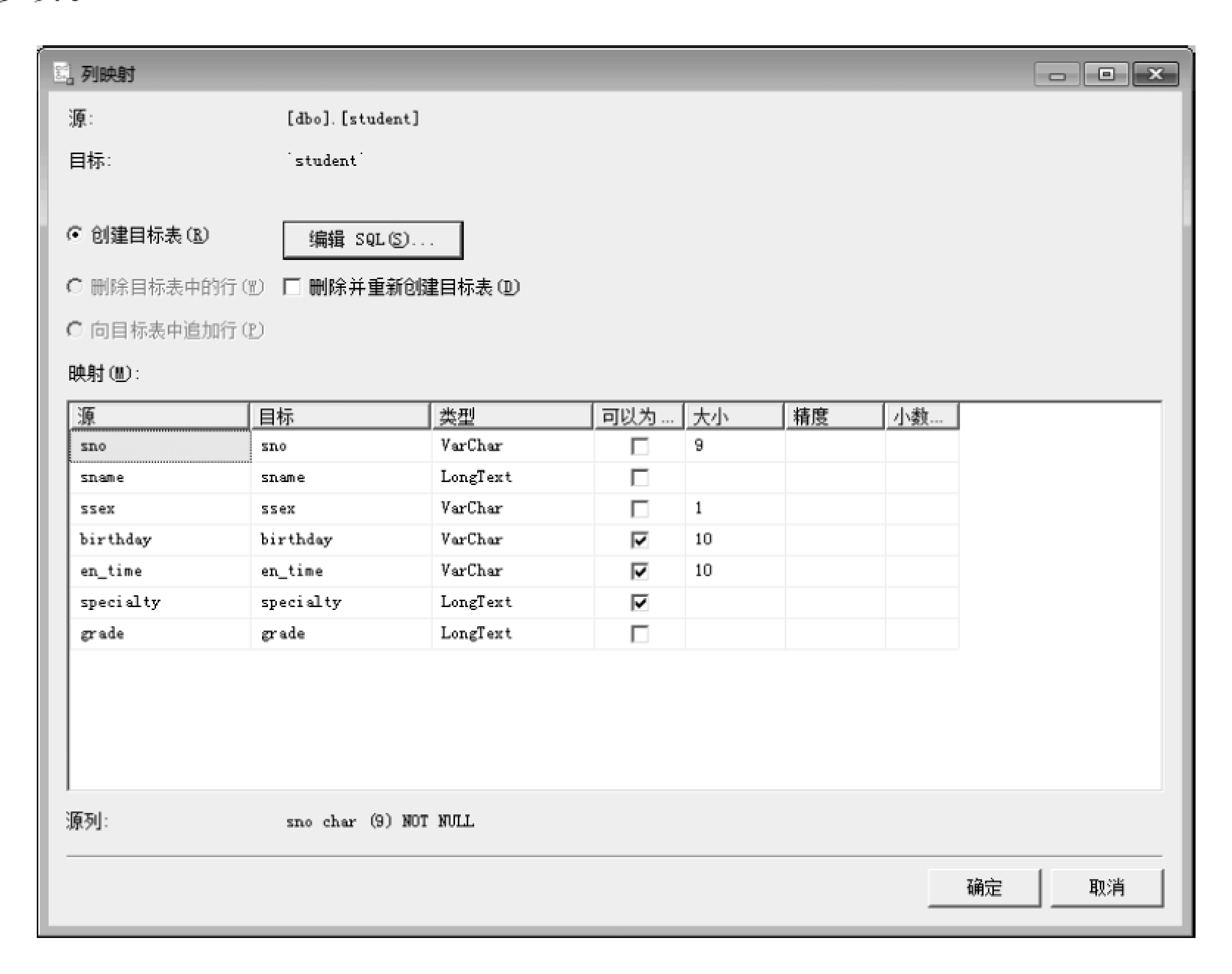


图 3-26 “列映射”窗口

⑦ 单击“确定”按钮，回到“选择源表和源视图”窗口，单击“下一步”按钮，出现“查看数据类型映射”窗口，如图 3-27 所示。



图 3-27 “查看数据类型映射”窗口

⑧ 单击“下一步”按钮，出现“运行包”窗口，如图 3-28 所示。



图 3-28 “运行包”窗口

⑨ 单击“下一步”按钮，出现“完成该向导”窗口，如图 3-29 所示。



图 3-29 “完成该向导”窗口

⑩ 单击“完成”按钮，出现“执行成功”窗口，如图 3-30 所示。单击“关闭”按钮即可。



图 3-30 “执行成功”窗口

打开“G:\data\st. xls”文件，查看导出的结果，如图 3-31 所示。



图 3-31 st. xls 文件

3.6.2 导入数据

导入数据是将其他格式的数据(如文本数据、Access、Excel、FoxPro 等)导入到 SQL Server 数据库中。

【例 3-33】 将一个 Excel 文件导入到 Student1 数据库中。

具体操作步骤：

① 在"对象资源管理器"中,展开数据库,右击要导入数据的数据库 Student1,弹出快捷菜单,选择"任务"级联菜单中的"导入数据"命令,如图 3-32 所示;出现"SQL Server 导入和导出向导"窗口,如图 3-21 所示。



图 3-32　选择"导入数据"命令

② 单击"下一步"按钮,出现"选择数据源"窗口,如图 3-33 所示,选择文件路径和文件格式。



图 3-33　"选择数据源"窗口

③ 单击“下一步”按钮，出现“选择目标”窗口，如图 3-34 所示，选择数据库。



图 3-34 “选择目标”窗口

④ 单击“下一步”按钮，出现“指定表复制或查询”窗口，如图 3-35 所示，选中“复制一个或多个表或视图的数据”单选按钮。



图 3-35 “指定表复制或查询”窗口

⑤ 单击“下一步”按钮，出现“选择源表和源视图”窗口，如图 3-36 所示，选择表或视图。



图 3-36 “选择源表和源视图”窗口

⑥ 单击“编辑映射”按钮，出现“列映射”窗口，如图 3-37 所示，可以在此对“列属性”等内容进行修改。



图 3-37 “列映射”窗口

⑦ 单击“确定”按钮，回到“选择源表和源视图”窗口，单击“下一步”按钮，出现“查看数据类型映射”窗口，如图 3-38 所示。



图 3-38 “查看数据类型映射”窗口

⑧ 单击“下一步”按钮，出现“运行包”窗口，如图 3-39 所示。



图 3-39 “运行包”窗口

⑨ 单击“下一步”按钮，出现“完成该向导”窗口，如图 3-40 所示。



图 3-40 “完成该向导”窗口

⑩ 单击“完成”按钮，出现“执行成功”窗口，如图 3-41 所示。单击“关闭”按钮，数据导入完成。



图 3-41 “执行成功”窗口

打开 Student1 数据库，查看导入的结果，如图 3-42 所示。

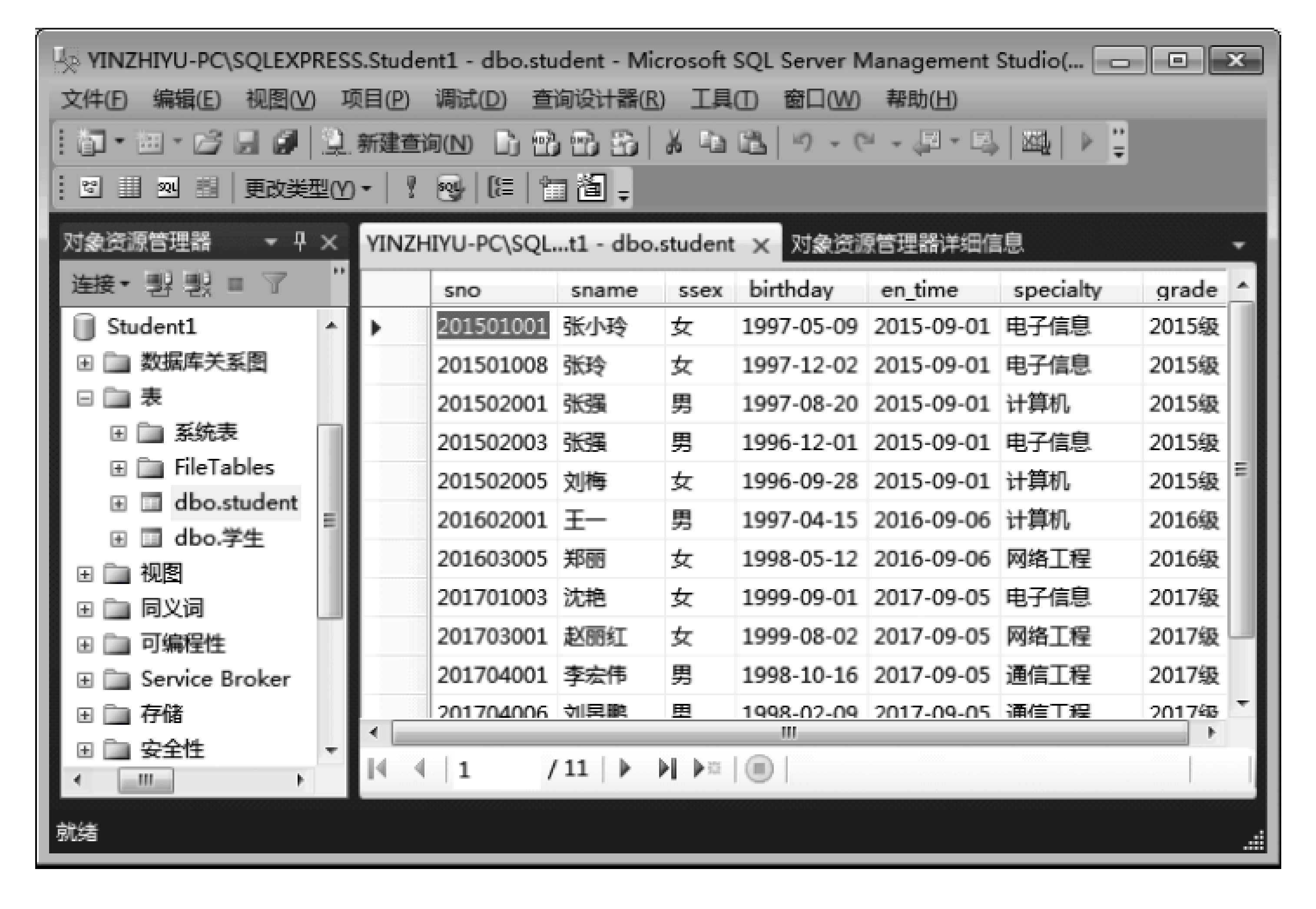


图 3-42　Student1 数据库中的 student 表

其他数据源和目标之间的数据导入和导出操作与上面的步骤基本相同，读者可以根据需要自行完成，这里不再讲述。

习　题　3

1. 利用 T-SQL 语句在第 2 章习题创建的 bankcard 数据库中完成下列操作。

(1) 创建 depositor 储户表，表结构如表 3-4 所示。

表 3-4　储户表的表结构

属性名	类型	宽度	键值	允许空否	取值范围	属性的含义
IDNO	char	18	主码	否		身份证号
Dname	nvarchar	10		否		姓名
Telephone	char	11		否	数字字符	手机号
VIP	nchar	1		否	是或否	贵宾储户

(2) 创建 account 账户表，表结构如表 3-5 所示。

表 3-5　账户表的表结构

属性名	类型	宽度	键值	允许空否	取值范围	属性的含义
AccNO	char	20	主码	否	数字字符	账号
IDNO	char	18	外键	否	参考储户表主码	身份证号
Password	char	6		否	数字字符	密码
OpenDate	date			否	默认值：当前日期	开户日期
CardType	nchar	3		否	信用卡、借记卡等	卡类型
MoneyType	nvarchar	10		否	人民币、美元等	币种
Balance	money			否		余额
ExpiryDate	date			否	＞OpenDate	有效期

(3) 创建 Trecord 交易记录表,表结构如表 3-6 所示。

表 3-6 交易记录表的表结构

属性名	类型	宽度	键值	允许空否	取值范围	属性的含义
ID	int		主码	否	IDENTITY(1,1),从 1 开始自动编号	顺序号
TDate	date			否	默认值:当前日期	交易时间
AccNO	char	20	外键	否	数字字符,参考账户表主码	账号
Expense	money			是		支出
Income	money			是		收入
OppAccNO	char	20	外键	是	数字字符	对方账号
Place	nvarchar	30		是		交易地点
Abstract	nvarchar	20		是	转账、消费、工资、ATM 取款等	摘要

2. 建立储户表、账户表和交易记录表三个表之间的关系图。

3. 分别给储户表、账户表和交易记录表添加数据如图 3-43、图 3-44 和图 3-45 所示。

IDNO	Dname	Telephone	VIP
121***195809250110	李华	13988880000	是
121***196809030111	郑洪洋	18199990000	否
130***197312120120	刘小钰	13066660000	否
130***198003090928	张伟	13933330000	是
133***198708150101	张萌萌	13011110000	是

图 3-43 储户表数据

AccNO	IDNO	Password	OpenDate	CardType	MoneyType	Balance	ExpiryDate
4125428003351201000 0	133***198708150101	555555	2016-09-01	信用卡	人民币	19000.0000	2021-09-01
41254280033512020000	130***197312120120	666666	2016-09-20	信用卡	人民币	0.0000	2021-09-01
41254280033512060000	130***198003090928	222222	2016-08-15	信用卡	人民币	12000.5500	2021-08-01
41254280033512080000	121***196809030111	654321	2016-08-13	信用卡	人民币	4367.1200	2021-08-01
43674280033512050000	133***198708150101	333333	2016-08-15	借记卡	人民币	2309.0000	2026-08-01
43674280033512070000	130***197312120120	111111	2016-09-12	借记卡	人民币	9878.3300	2026-09-01
43674280033512090000	121***195809250110	123456	2016-08-13	借记卡	人民币	5678.0900	2026-08-01

图 3-44 账户表数据

ID	TDate	AccNO	Expense	Income	OppAccNO	Place	Abstract
1	2016-08-20	41254280033512060000	600.0000	NULL	41254280033512080000	友谊大街分行	转账支出
2	2016-08-20	41254280033512080000	103.0000	NULL	43674280033512050000	北国超市	消费支出
3	2016-08-20	41254280033512080000	NULL	600.0000	41254280033512060000	友谊大街分行	转账收入
4	2016-08-20	43674280033512050000	NULL	103.0000	41254280033512080000	北国超市	销售收入
5	2016-08-23	43674280033512070000	125.0000	NULL	43674280033512090000	新华书店	消费支出
6	2016-08-23	43674280033512090000	NULL	125.0000	43674280033512070000	新华书店	销售收入
7	2016-08-25	43674280033512090000	NULL	500.0000	NULL	槐北路支行	ATM存款
8	2016-08-25	43674280033512050000	8000.0000	NULL	NULL	槐安路分行	ATM取款
9	2016-08-26	43674280033512070000	NULL	200.0000	43674280033512050000	NULL	防暑费收入
10	2016-08-26	43674280033512050000	200.0000	NULL	43674280033512070000	NULL	防暑费支出
11	2016-08-26	43674280033512090000	800.0000	NULL	43674280033512070000	NULL	转账支出
12	2016-08-26	43674280033512070000	NULL	800.0000	43674280033512090000	NULL	转账收入
13	2016-08-28	41254280033512060000	NULL	6780.0000	43674280033512050000	NULL	9月份工资收入
14	2016-08-28	43674280033512050000	6780.0000	NULL	41254280033512060000	NULL	9月份工资支出

图 3-45 交易记录表数据

4. 练习数据库表中数据的导入和导出。

第 4 章　数据库查询

所谓查询,就是检索数据库内数据的特定请求。数据库接受用 T-SQL 语言编写的查询,使用查询可以按照不同的方式查看和分析数据。查询设计是数据库应用程序开发的重要组成部分,因为在设计好数据库并用数据进行填充后,最常见的就是通过查询来使用数据。

本章主要介绍数据库的基本查询,包括投影、选择、聚合等简单查询,分组及数据汇总查询,内连接、外连接等连接查询,无关及相关子查询,对查询结果排序等等。

4.1　SELECT 查询语法

在 SQL Server 中,可以通过 SELECT 语句来实现数据查询,即从数据库表中检索所需要的数据。查询可以包含要返回的列、要选择的行、放置行的顺序以及如何将信息分组的规范。

SELECT 语句的基本语法格式如下:

```
SELECT { select_list [INTO new_table_name ]}
FROM { table_list | view_list}
[ WHERE {search_conditions} ]
[ GROUP BY { group_by_list} ]
[ HAVING { search_conditions} ]
[ ORDER BY { order_list [ ASC | DESC ] }]
```

其中各参数的含义如下:

① select_list——描述结果集的列,它指定了结果集中要包含的列的名称,是一个逗号分隔的表达式列表。

② INTO new_table_name——指定使用结果集来创建新表。new_table_name 指定新表的名称。

③ FROM {table_list|view_list }——指定要从中检索数据的表名或视图名。

④ WHERE{search_conditions}——WHERE 子句是一个筛选条件,它定义了源表中的行要满足 SELECT 语句的要求所必须达到的条件。

⑤ GROUP BY{group_by_list}——GROUP BY 子句根据 group_by_list 列中的值将结果集分成组。

⑥ HAVING{search_conditions}——HAVING 子句应用于结果集中组的附加筛选,用来向使用 GROUP BY 子句的查询中添加数据过滤准则。

⑦ ORDER BY{order_list[ASC | DESC]}——ORDER BY 子句定义了结果集中行的排序顺序。升序使用 ASC 关键字,降序使用 DESC 关键字,默认为升序。

以上为 SELECT 语句的基本语法,只包含了主要查询功能。有些功能的语法和应用会在后面的章节进行详细介绍。

4.2 简单查询

简单查询包括投影查询、选择查询及采用聚合函数的简单计算查询。

4.2.1 投影查询

通过 SELECT 语句的 select_list 项组成结果表的列。

投影查询格式如下:

```
SELECT [ ALL | DISTINCT ] [ TOP n [ PERCENT ] ]
{ * | { {column_name | expression | IDENTITYCOL | ROWGUIDCOL }
    [ [ AS ] column_alias ] | column_alias = expression } [, … n ] }
```

其中参数的含义如下:

① ALL——指定显示所有记录,包括重复行。ALL 是默认设置。

② DISTINCT——指定显示所有记录,但不包括重复行。

③ TOP n [PERCENT]——指定从查询结果中返回前 n 行或前百分之 n 行。

④ *——表示所有列。

⑤ column_name——指定要返回的列名。

⑥ expression——是列名、常量、函数以及由运算符连接的列名、常量和函数的任意组合,或者是子查询。

⑦ column_alias——列别名。

1. 选择一个表中指定的列

使用 SELECT 语句选择一个表中的某些列,各列名之间要以逗号分隔。

【例 4-1】 查询 teaching 数据库中学生的姓名、性别和专业。

```
USE teaching
SELECT sname,ssex, specialty FROM student
```

查询结果如图 4-1 所示。

【例 4-2】 查询 teaching 数据库中 course 的所有记录。

```
USE teaching
SELECT * FROM course
```

用“*”表示表中所有的列,按用户创建表时声明列的顺序来显示所有的列。

【例 4-3】 查询 teaching 中“学生表”的专业名称,滤掉重复行。

```
USE teaching
SELECT DISTINCT specialty FROM student
```

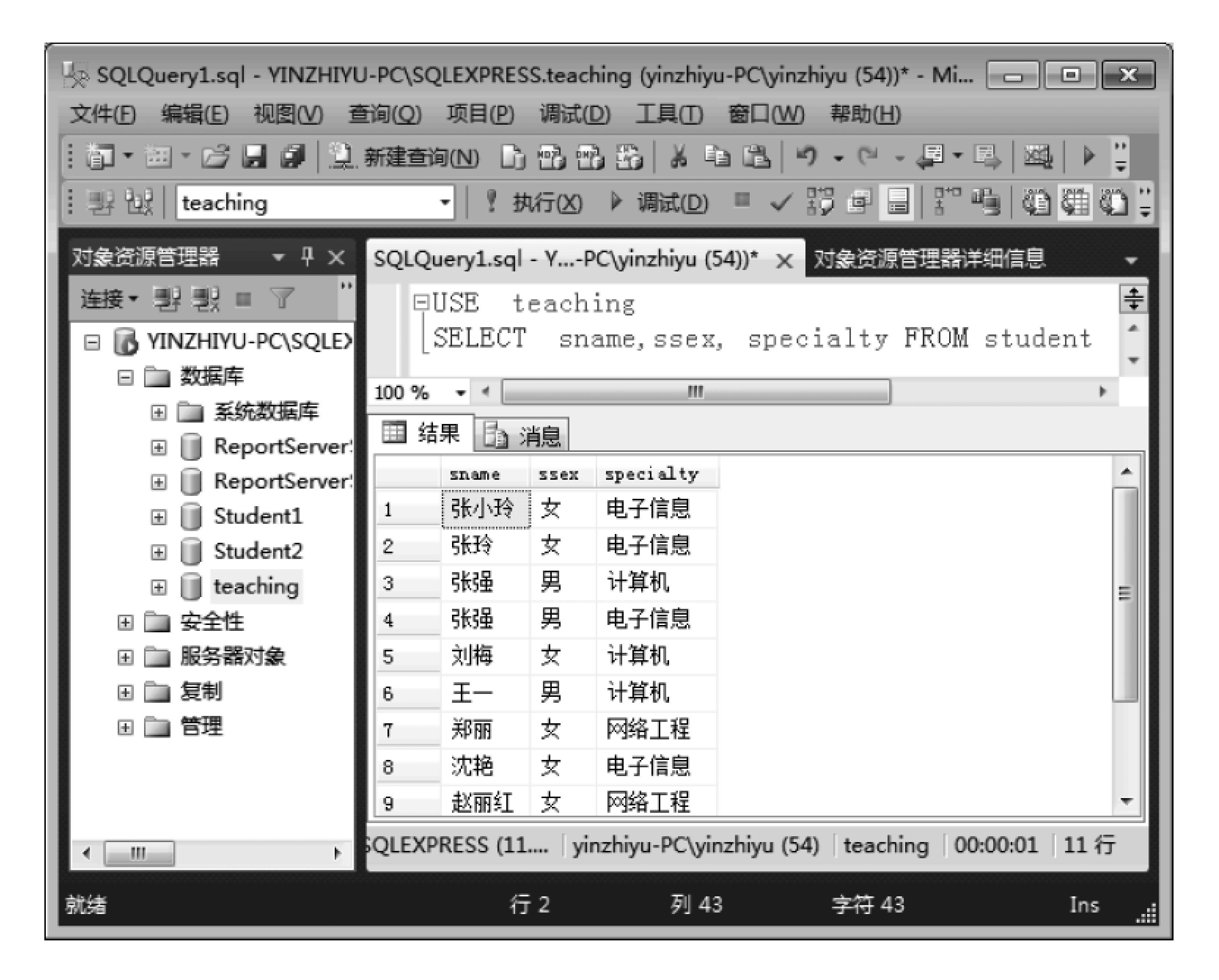


图 4-1　例 4-1 的查询结果

用 DISTINCT 关键字可以过滤掉查询结果中的重复行。

【例 4-4】　查询 teaching 库中 course 表的前三行信息。

```
USE  teaching
SELECT  top 3 * FROM course
```

查询结果如图 4-2 所示。

【例 4-5】　查询 teaching 库中 course 表的前 50%行的信息。

```
USE  teaching
SELECT top 50 percent * FROM course
```

查询结果如图 4-3 所示。



图 4-2　例 4-4 的查询结果

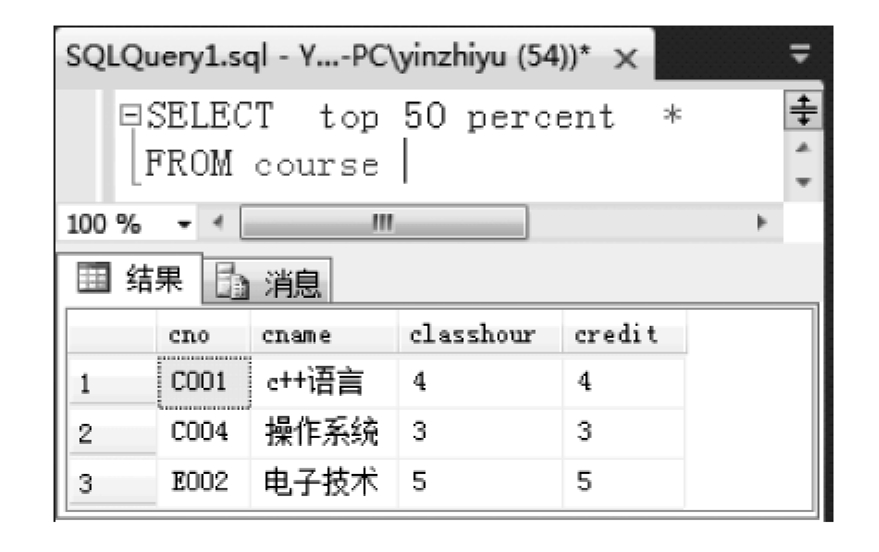


图 4-3　例 4-5 的查询结果

2．改变查询结果中的显示标题

在 SELECT 语句中，用户可以根据实际需要对查询数据的列标题进行改变，或者为没有标题的列加上临时的标题。

常用的方式：

① 在列表达式后面给出列名。

② 用“=”来连接列表达式。

③ 用 AS 关键字来连接列表达式和指定的列名。

【例 4-6】 查询 student 表中所有学生的学号、姓名，结果中各列的标题分别指定为“学号”和“姓名”。

```
USE  teaching
SELECT sno AS 学号, sname AS
姓名 FROM student
```

查询结果如图 4-4 所示。

或

```
USE  teaching
SELECT 学号 = sno,姓名 = sname FROM student
```

或

```
USE  teaching
SELECT sno 学号, sname 姓名 FROM student
```



	学号	姓名
1	201501001	张小玲
2	201501008	张玲
3	201502001	张强
4	201502003	张强
5	201502005	刘梅
6	201602001	王一
7	201603005	郑丽

图 4-4　例 4-6 的查询结果

注意：列标题别名只在定义的语句中有效，即只是显示标题，对原表中的列标题没有任何影响。

3. 计算列值

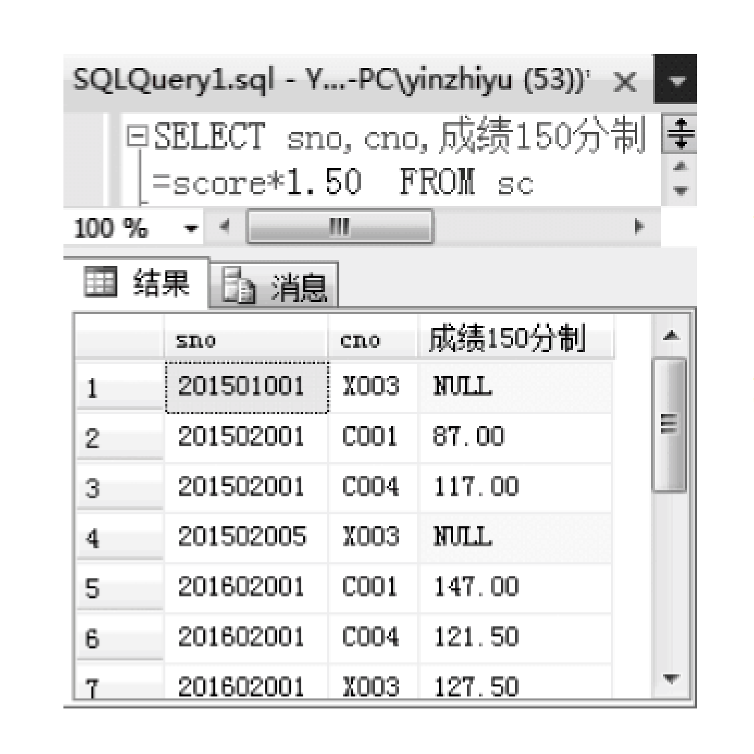
在进行数据查询时，经常需要对查询到的数据进行再次计算处理。

T-SQL 允许直接在 SELECT 语句中使用计算列。“计算列”并不存在于表格所存储的数据中，它是通过对某些列的数据进行演算得来的结果，所以没有列名。

【例 4-7】 查询 sc 表，按 150 分制计算成绩。

```
USE  teaching
SELECT sno,cno,成绩 150 分制 = score * 1.50  FROM sc
```

查询结果如图 4-5 所示。



	sno	cno	成绩150分制
1	201501001	X003	NULL
2	201502001	C001	87.00
3	201502001	C004	117.00
4	201502005	X003	NULL
5	201602001	C001	147.00
6	201602001	C004	121.50
7	201602001	X003	127.50

图 4-5　例 4-7 的查询结果

4.2.2　选择查询

投影查询是从列的角度进行的查询，一般对行不进行任何过滤（DISTINCT 除外）。但是，一般的查询都不是针对全表所有行的查询，只是从整个表中选出满足指定条件的内容，这就要用到 WHERE 子句进行选择查询。

选择查询的基本语法如下：

```
SELECT  SELECT_LIST
FROM   TABLE_LIST
WHERE SEARCH_CONDITIONS
```

其中，SEARCH_CONDITIONS 为选择查询的条件。

SQL Server 支持比较、范围、列表、字符串匹配等选择方法。

WHERE 子句中常用的条件表达式见表 4-1。SQL Server 对 WHERE 子句中的查询条件的数目没有限制。

表 4-1 常用的查询条件

查询条件	谓词
比较运算符	=，>，<，>=，<=，!=，<>，!>，!<
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件	AND，OR，NOT

1. 使用关系表达式

比较运算符用于比较两个表达式的值，共有 9 个，它们是：=（等于）、<（小于）、<=（小于或等于）、>（大于）、>=（大于或等于）、<>（不等于）、!=（不等于）、!<（不小于）、!>（不大于）。

比较运算的格式为：

```
expression { = | < | <= | > | >= | <> | != | !< | !> } expression
```

其中 expression 是除 text、ntext 和 image 外类型的表达式。

【例 4-8】 查询 teaching 数据库 sc 表中成绩大于或等于 60 的学生的学号、课程号和成绩。

```
USE  teaching
SELECT * FROM sc WHERE score>=60
```

2. 使用逻辑表达式

逻辑运算符共有 3 个，它们是：

NOT——非，对表达式的否定。

AND——与，连接多个条件，所有的条件都成立时为真。

OR——或，连接多个条件，只要有一个条件成立就为真。

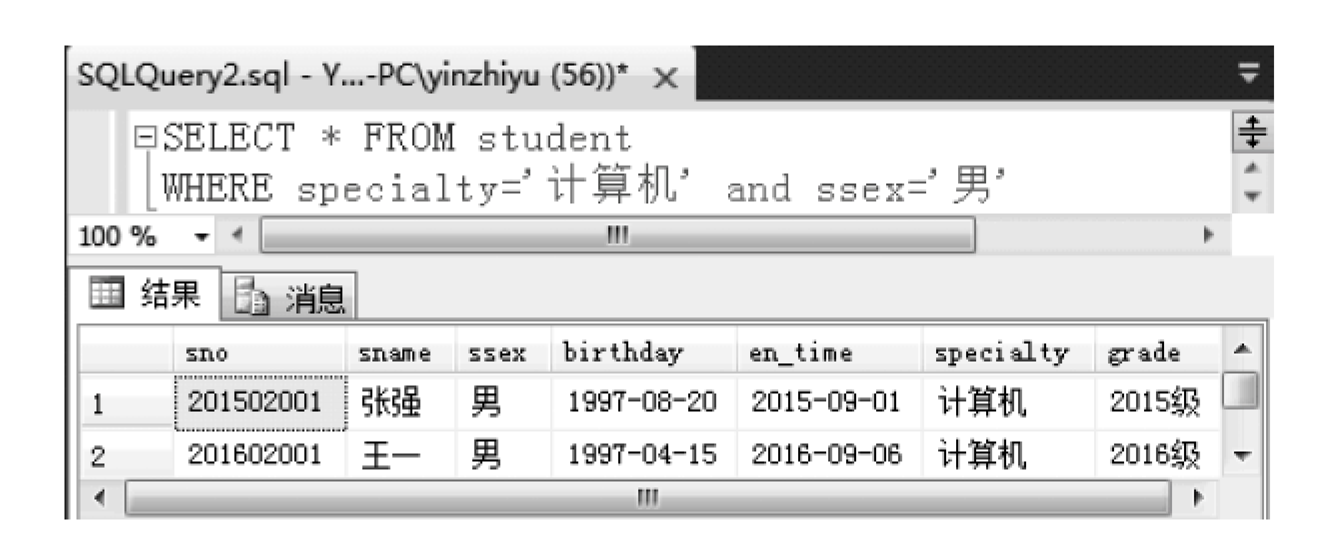
【例 4-9】 查询 teaching 数据库中“计算机”专业的“男”生的信息。

```
USE  teaching
SELECT * FROM student
WHERE specialty='计算机' and ssex='男'
```

查询结果如图 4-6 所示。

【例 4-10】 查询 teaching 数据库中“计算机”专业的或“男”生的信息。

```
USE  teaching
SELECT * FROM student
WHERE specialty='计算机' or  ssex='男'
```



	sno	sname	ssex	birthday	en_time	specialty	grade
1	201502001	张强	男	1997-08-20	2015-09-01	计算机	2015级
2	201602001	王一	男	1997-04-15	2016-09-06	计算机	2016级

图 4-6　例 4-9 的查询结果

查询结果如图 4-7 所示。

SQLQuery2.sql - Y...-PC\yinzhiyu (56))*

```
SELECT * FROM student
WHERE specialty='计算机' or  ssex='男'
```

	sno	sname	ssex	birthday	en_time	specialty	grade
1	201502001	张强	男	1997-08-20	2015-09-01	计算机	2015级
2	201502003	张强	男	1996-12-01	2015-09-01	电子信息	2015级
3	201502005	刘梅	女	1996-09-28	2015-09-01	计算机	2015级
4	201602001	王一	男	1997-04-15	2016-09-06	计算机	2016级
5	201704001	李宏伟	男	1998-10-16	2017-09-05	通信工程	2017级
6	201704006	刘景鹏	男	1998-02-09	2017-09-05	通信工程	2017级

图 4-7　例 4-10 的查询结果

3. 使用 BETWEEN 关键字

使用 BETWEEN 关键字可以更方便地限制查询数据的范围。其语法格式为：

```
表达式 [NOT] BETWEEN  表达式 1 AND 表达式 2
```

使用 BETWEEN 表达式进行查询的效果完全可以用含有“>=”和“<=”的逻辑表达式来代替，使用 NOT BETWEEN 进行查询的效果完全可以用含有“>”和“<”的逻辑表达式来代替。

【例 4-11】　查询 teaching 数据库中成绩在 80～90 的学生的学号、课程号和成绩。

```
USE  teaching
SELECT * FROM sc
WHERE score BETWEEN 80 AND 90
```

或

```
USE  teaching
SELECT * FROM sc
WHERE score>= 80 AND score<=90
```

查询结果如图 4-8 所示。

SQLQuery2.sql - Y...-PC\yinzhiyu (56))*

```
SELECT * FROM sc
WHERE score BETWEEN 80 AND 90
```

	sno	cno	score
1	201602001	C004	81
2	201602001	X003	85
3	201703001	C001	85

图 4-8　例 4-11 的查询结果

【例 4-12】　查询 teaching 数据库中成绩不在 80～90 的学生的学号、课程号和成绩。

```
USE  teaching
SELECT * FROM sc
```

```
WHERE score NOT BETWEEN 80 AND 90
```

4. 使用 IN(属于)关键字

同 BETWEEN 关键字一样,IN 的引入也是为了更方便地限制检索数据的范围。

语法格式为:

```
表达式 [NOT] IN (表达式 1 , 表达式 2 [,…,表达式 n])
```

【例 4-13】 查询 teaching 数据库中"计算机"和"通信工程"专业的学生的姓名、学号和专业。

```
USE   teaching
SELECT sname,sno,specialty FROM student
WHERE specialty IN('计算机','通信工程')
```

5. 使用 LIKE 关键字

使用 LIKE 关键字的查询又叫模糊查询,LIKE 关键字搜索与指定模式匹配的字符串、日期或时间值。字符串中可包含 4 种通配符的任意组合,搜索条件中可用的通配符如表 4-2 所示。

表 4-2 常用的通配符

通配符	含 义
%	包含零个或多个字符的任意字符串
_	任何单个字符
[ ]	代表指定范围内的单个字符,[]中可以是单个字符(如[acef]),也可以是字符范围(如[a-f])
[^]	代表不在指定范围内的单个字符,[^]中可以是单个字符(如[^acef]),也可以是字符范围(如[^a-f])

【例 4-14】 通配符的示例。

LIKE 'AB%'返回以 AB 开始的任意字符串。

LIKE 'Ab%'返回以 Ab 开始的任意字符串。

LIKE '%abc'返回以 abc 结束的任意字符串。

LIKE '%abc%'返回包含 abc 的任意字符串。

LIKE '_ab'返回以 ab 结束的三个字符的字符串。

LIKE '[ACK]% '返回以 A、C 或 K 开始的任意字符串。

LIKE '[A-T]ing'返回四个字符的字符串,结尾是 ing,首字符的范围从 A 到 T。

LIKE 'M[^c]% '返回以 M 开始且第二个字符不是 c 的任意长度的字符串。

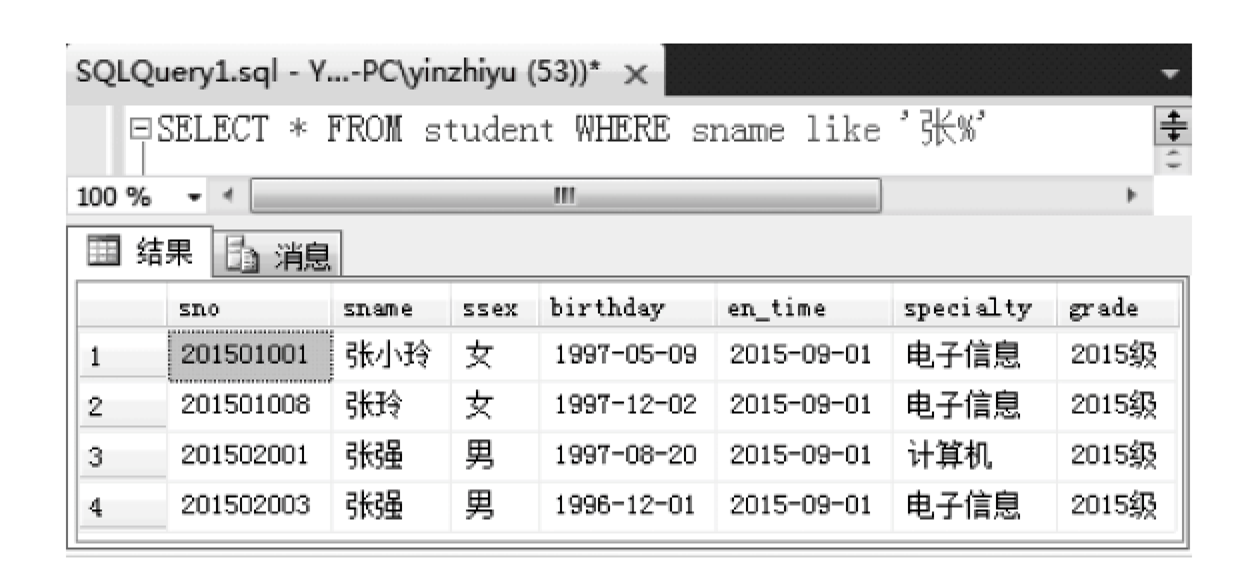
【例 4-15】 查询 teaching 数据库中所有姓"张"的学生的信息。

```
USE teaching
SELECT * FROM student
WHERE sname like '张%'
```

查询结果如图 4-9 所示。

6. IS [NOT]NULL(是[否]为空)查询

在 WHERE 子句中不能使用比较运算符对空值进行判断,只能使用空值表达式来判断


SQLQuery1.sql - Y...-PC\yinzhiyu (53))*

```
SELECT * FROM student WHERE sname like '张%'
```

	sno	sname	ssex	birthday	en_time	specialty	grade
1	201501001	张小玲	女	1997-05-09	2015-09-01	电子信息	2015级
2	201501008	张玲	女	1997-12-02	2015-09-01	电子信息	2015级
3	201502001	张强	男	1997-08-20	2015-09-01	计算机	2015级
4	201502003	张强	男	1996-12-01	2015-09-01	电子信息	2015级

图 4-9　例 4-15 的查询结果

某个列值是否为空值。

语法格式：

```
表达式 IS [NOT] NULL
```

【例 4-16】　查询 teaching 数据库中所有“成绩”为空值的学生的学号、课程号和成绩。

```
USE teaching
SELECT * FROM sc WHERE score IS NULL
```

7. 复合条件查询

在 WHERE 子句中可以使用逻辑运算符把若干个搜索条件合并起来，组成复杂的复合搜索条件。这些逻辑运算符包括 AND、OR 和 NOT。

AND 运算符：表示只有在所有条件都为真时，才返回真。

OR 运算符：表示只要有一个条件为真，就可以返回真。

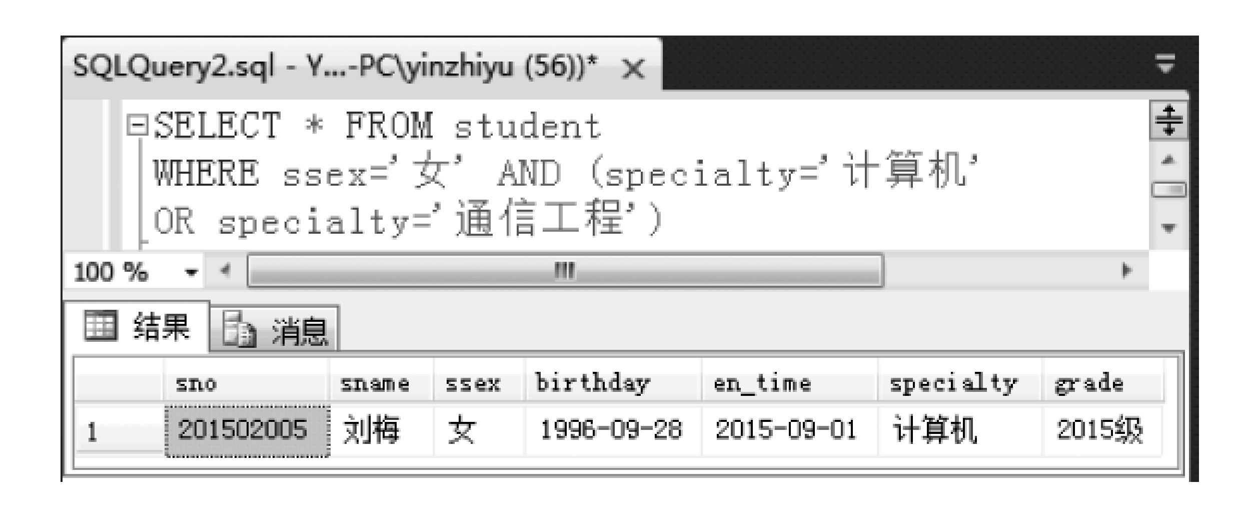
NOT 运算符：取反。

当在一个 WHERE 子句中同时包含多个逻辑运算符时，其优先级从高到低依次是 NOT、AND、OR。

【例 4-17】　从 teaching 数据库的 student 表中查询所有“计算机”和“通信工程”专业的“女”生的信息。

```
USE teaching
SELECT * FROM student
WHERE ssex = '女' AND
(specialty = '计算机'
OR specialty = '通信工程')
```

查询结果如图 4-10 所示。


SQLQuery2.sql - Y...-PC\yinzhiyu (56))*

```
SELECT * FROM student
WHERE ssex='女' AND (specialty='计算机'
OR specialty='通信工程')
```

	sno	sname	ssex	birthday	en_time	specialty	grade
1	201502005	刘梅	女	1996-09-28	2015-09-01	计算机	2015级

图 4-10　例 4-17 的查询结果

4.2.3 聚合函数查询

SQL Server 提供了一系列聚合函数。这些函数把存储在数据库中的数据描述为一个整体而不是一行行孤立的记录，通过使用这些函数可以实现数据集合的汇总或是求平均值等各种运算。

T-SQL 提供聚合函数如表 4-3 所示。

表 4-3 常用的聚合函数

函数名	功　能
sum(列名)	返回一个数字列的总和
avg(列名)	对一个数字列计算平均值
min(列名)	返回一个数字、字符串或日期列的最小值
max(列名)	返回一个数字、字符串或日期列的最大值
count(列名)	返回一个列的数据项数
count(*)	返回找到的行数

在 SELECT 子句中可以使用聚合函数进行运算，运算结果作为新列出现在结果集中，但此列无列名。在聚合运算的表达式中，可以包括列名、常量以及由算术运算符连接起来的函数。

【例 4-18】 在 teaching 数据库中查询 sc 表中成绩的平均值，平均值显示列标题为“平均成绩”。

```
USE  teaching
SELECT avg(score) AS 平均成绩 FROM sc
```

查询结果如图 4-11 所示。

【例 4-19】 从 teaching 数据库的 student 表中查询专业的种类个数（相同的按一种计算）。

```
USE  teaching
SELECT count (DISTINCT specialty) AS 专业种类数
FROM student
```

查询结果如图 4-12 所示。

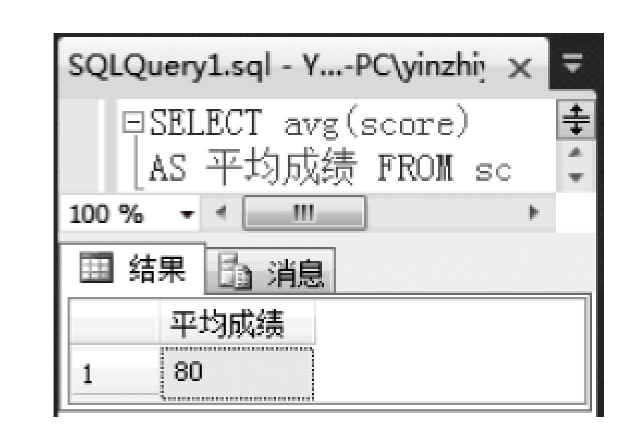


图 4-11 例 4-18 的查询结果

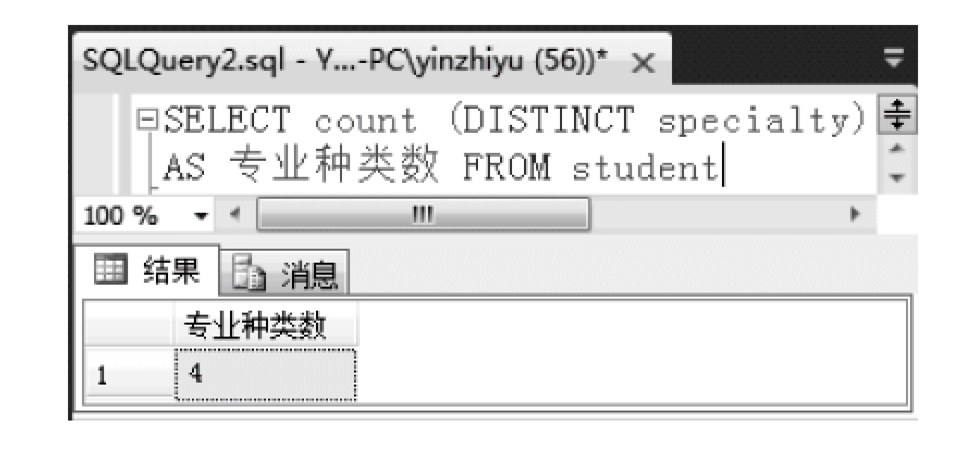


图 4-12 例 4-19 的查询结果

说明：在 T-SQL 中，允许与统计函数如 count()、sum()和 avg()一起使用 DISTINCT 关键字来处理列或表达式中不同的值。

【例 4-20】 在 teaching 数据库中查询学号为 201501001 的学生平均成绩和最高成绩。

```
USE  teaching
SELECT avg(score) AS 平均成绩, max(score) AS 最高成绩 FROM sc
WHERE sno = '201501001'
```

在 Microsoft SQL Server 2008 系统中,一般情况下可以在两个地方使用聚合函数,即 SELECT 子句和 HAVING 子句。

4.3 分组查询

使用聚合函数返回的是所有行数据的统计结果。如果需要按某一列数据的值进行分类,在分类的基础上再进行查询,就要使用 GROUP BY 子句了。分组技术是指使用 GROUP BY 子句完成分组操作的技术。

GROUP BY 子句的语法结构如下:

```
[ GROUP BY {[ ALL ] group_by_expression [,…n]}
[ WITH { CUBE | ROLLUP } ] ]
```

其中的参数说明如下:

① ALL——包含所有的组和结果,甚至包含那些不满足 WHERE 子句指定搜索条件的组和结果。如果指定了 ALL,那么组中不满足搜索条件的空值也将作为一个组。

② group_by_expression——执行分组的表达式,可以是列或引用列的非聚合表达式。

③ CUBE——除了返回由 GROUP BY 子句指定的列外,还返回按组统计的行,返回的结果先按分组的第一个条件列排序显示,再按第二个条件列排序显示,以此类推,统计行包括了 GROUP BY 子句指定的列的各种组合的数据统计,更改列分组的顺序会影响在结果集内生成的行数。

④ ROLLUP——此选项只返回最高层的分组列即第一个分组列的统计数据。

4.3.1 简单分组

如果在 GROUP BY 子句中没有使用 CUBE 或 ROLLUP 关键字,那么表示这种分组的技术是简单分组技术。

【例 4-21】 查询 teaching 数据库中男生和女生的人数。

```
USE  teaching
SELECT ssex, count(ssex) 人数
FROM student
GROUP BY ssex
```

查询结果如图 4-13 所示。

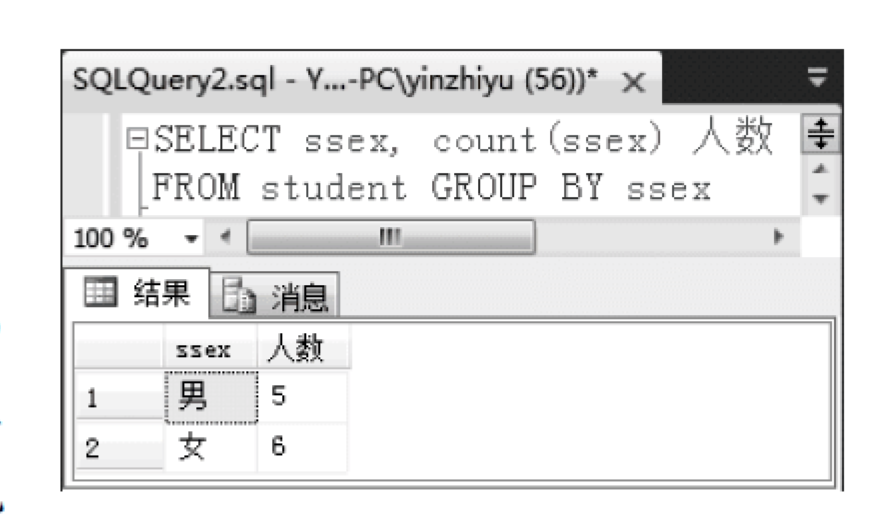


图 4-13 例 4-21 的查询结果

注意: 使用 GROUP BY 子句时,选择(SELECT)列表中任何非聚合表达式内的所有列都应包含在 GROUP BY 列表中(不能使用列别名),或者说 GROUP BY 表达式必须与选择(SELECT)列表表达

式完全匹配。

当完成数据结果的查询和统计后，可以使用 HAVING 关键字来对查询和统计的结果进行进一步的筛选。

【例 4-22】 在 sc 表中查询选修了两门及以上课程的学生学号和选课数。

```
USE  teaching
SELECT sno,COUNT(cno) 选修课程数 FROM sc
GROUP BY sno
HAVING COUNT(cno)> = 2
```

查询结果如图 4-14 所示。

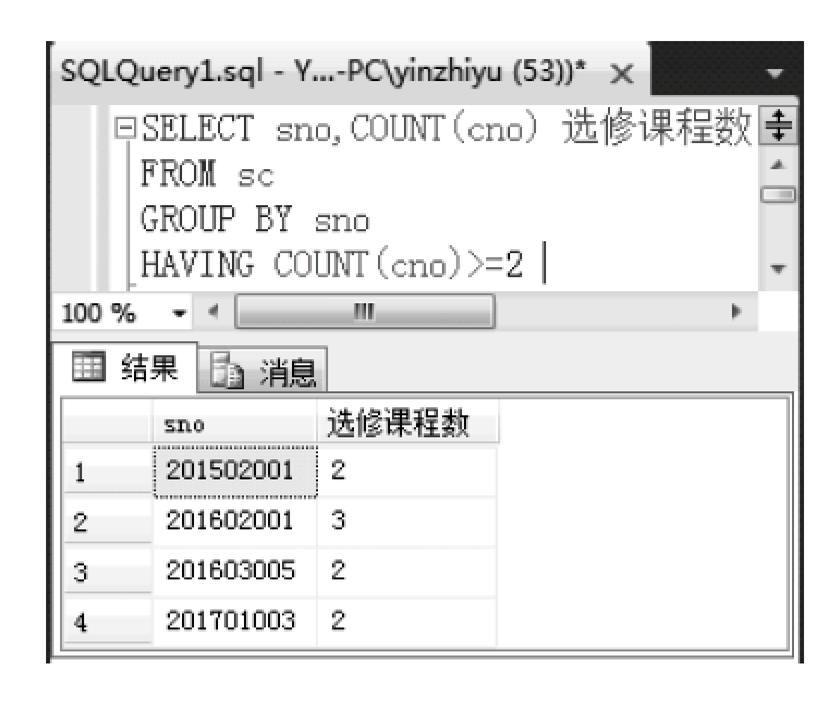


图 4-14　例 4-22 的查询结果

HAVING 与 WHERE 子句的区别：WHERE 子句是对整表(源表)中数据筛选满足条件的行；而 HAVING 子句是对 GROUP BY 分组查询后产生的组再加条件，筛选出满足条件的组。另外，HAVING 中的条件一般都直接使用聚合函数，WHERE 中的条件不能直接使用聚合函数。

4.3.2　CUBE 和 ROLLUP 的应用

1. CUBE

指定在结果集内不仅包含由 GROUP BY 提供的行，还包含汇总行。GROUP BY 汇总行在结果集内返回每个可能的组和子组组合。GROUP BY 汇总行在结果中显示为 NULL，但用来表示所有值。使用 GROUPING 函数可确定结果集内的空值是否为 GROUP BY 汇总值。

结果集内的汇总行数取决于 GROUP BY 子句内包含的列数。GROUP BY 子句中的每个操作数(列)绑定在分组 NULL 下，并且分组适用于所有其他操作数(列)。由于 CUBE 返回每个可能的组和子组组合，因此不论在列分组时指定使用什么顺序，行数都相同。

【例 4-23】 在 teaching 数据库中查询 sc 表，求被选修的各门课程的平均成绩和选修该课程的人数，及所有课程的总平均成绩和总选修人数。

```
USE teaching
SELECT cno, AVG(score) AS '平均成绩',COUNT(sno) AS
'选修人数'
FROM sc
GROUP BY cno
WITH CUBE
```

查询结果如图 4-15 所示。

```
SELECT cno, AVG(score) AS '平均成绩',
COUNT(sno) AS '选修人数'
FROM sc
GROUP BY cno
WITH CUBE
```

	cno	平均成绩	选修人数
1	C001	78	5
2	C004	85	3
3	E002	74	2
4	X003	85	3
5	NULL	80	13

图 4-15　例 4-23 的查询结果

【例 4-24】 在 teaching 数据库中查询 student 表，统计各专业男生、女生人数及每个专业的学生人数和男生总人数、女生人数以及所有学生总人数。

```
USE teaching
```

```
SELECT specialty,ssex,COUNT( * ) AS '人数'
FROM student
GROUP BY specialty,ssex
WITH CUBE
```

查询结果如图 4-16 所示。

2. ROLLUP

ROLLUP 指定在结果集内不仅包含由 GROUP BY 提供的行，还包含汇总行。按层次结构顺序，从组内的最低级别到最高级别汇总组。组的层次结构取决于列分组时指定使用的顺序。更改列分组的顺序会影响在结果集内生成的行数。

使用 CUBE 或 ROLLUP 时，不支持区分性聚合函数，如 AVG(DISTINCT 列名)、COUNT(DISTINCT 列名)等。

【例 4-25】 统计在 teaching 数据库中每个专业的男女生人数，每个专业的总人数和所有学生总人数。

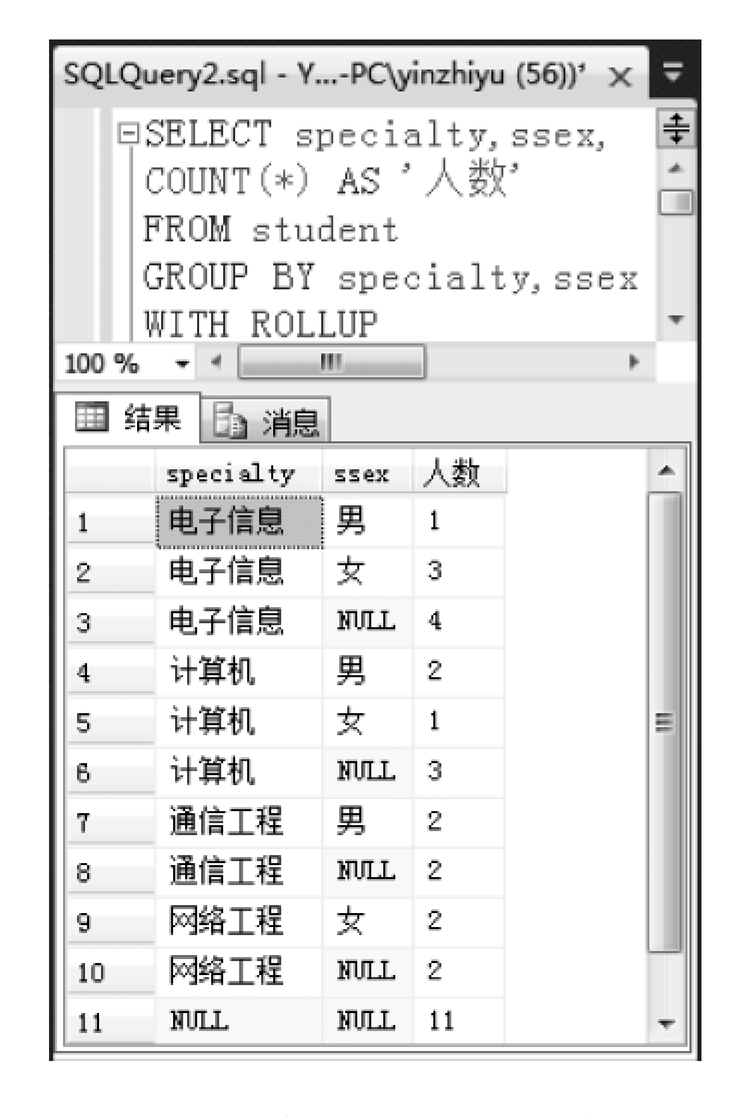
```
USE teaching
SELECT specialty,ssex,COUNT( * ) AS '人数'  FROM student
GROUP BY specialty,ssex
WITH ROLLUP
```

查询结果如图 4-17 所示。



	specialty	ssex	人数
1	电子信息	男	1
2	计算机	男	2
3	通信工程	男	2
4	NULL	男	5
5	电子信息	女	3
6	计算机	女	1
7	网络工程	女	2
8	NULL	女	6
9	NULL	NULL	11
10	电子信息	NULL	4
11	计算机	NULL	3
12	通信工程	NULL	2
13	网络工程	NULL	2

图 4-16　例 4-24 的查询结果



	specialty	ssex	人数
1	电子信息	男	1
2	电子信息	女	3
3	电子信息	NULL	4
4	计算机	男	2
5	计算机	女	1
6	计算机	NULL	3
7	通信工程	男	2
8	通信工程	NULL	2
9	网络工程	女	2
10	网络工程	NULL	2
11	NULL	NULL	11

图 4-17　例 4-25 的查询结果

4.4　连接查询

以上的查询操作都是从一个表中检索数据。在实际应用中，经常需要同时从两个表或两个以上表中检索数据，并且每一个表中的数据往往作为一个单独的列出现在结果集中。

实现从两个或两个以上表中检索数据且结果集中出现的列来自于两个或两个以上表中的检索操作被称为连接技术,或者说连接技术是指对两个表或两个以上表中数据执行乘积运算的技术。

在 Microsoft SQL Server 2008 系统中,这种连接操作又可以细分为内连接、自连接、外连接等。下面分别研究这些连接技术。

4.4.1 内连接

内连接把两个表中的数据连接生成第三个表,第三个表中仅包含那些满足连接条件的数据行。在内连接中,使用 INNER JOIN 连接运算符,并且使用 ON 关键字指定连接条件。

内连接是一种常用的连接方式,如果在 JOIN 关键字前面没有明确指定连接类型,那么默认的连接类型是内连接。内连接的语法格式如下:

```
SELECT   select_list
FROM   表 1 INNER JOIN 表 2 ON 连接条件
```

或

```
SELECT   select_list
FROM   表 1,表 2   WHERE 连接条件
```

连接条件格式为:

```
[<表名 1>.]<列名 1> <比较运算符> [<表名 2>.]<列名 2>
```

【例 4-26】 从 teaching 数据库中查询每个学生的姓名、课程号和成绩。

```
USE teaching
SELECT student.sname, sc.cno, sc.score
FROM student INNER JOIN sc
ON student.sno = sc.sno
```

也可以利用下面的程序来实现:

```
USE teaching
SELECT student.sname, sc.cno, sc.score
FROM student,sc
WHERE student.sno = sc.sno
```

查询结果如图 4-18 所示。

注意:当从多个表中查询的列名相同时,列名前必须加表名;列名不同时,列名前可以不加表名,但有时也会加上表名,以增强可读性。

【例 4-27】 从 teaching 数据库中查询“计算机”专业的学生所选的每门课的平均分。

```
USE teaching
SELECT b.cno, avg(b.score) as 平均分
FROM student a   INNER JOIN sc b
ON a.sno = b.sno and a.specialty = '计算机'
GROUP BY b.cno
```

查询结果如图 4-19 所示。

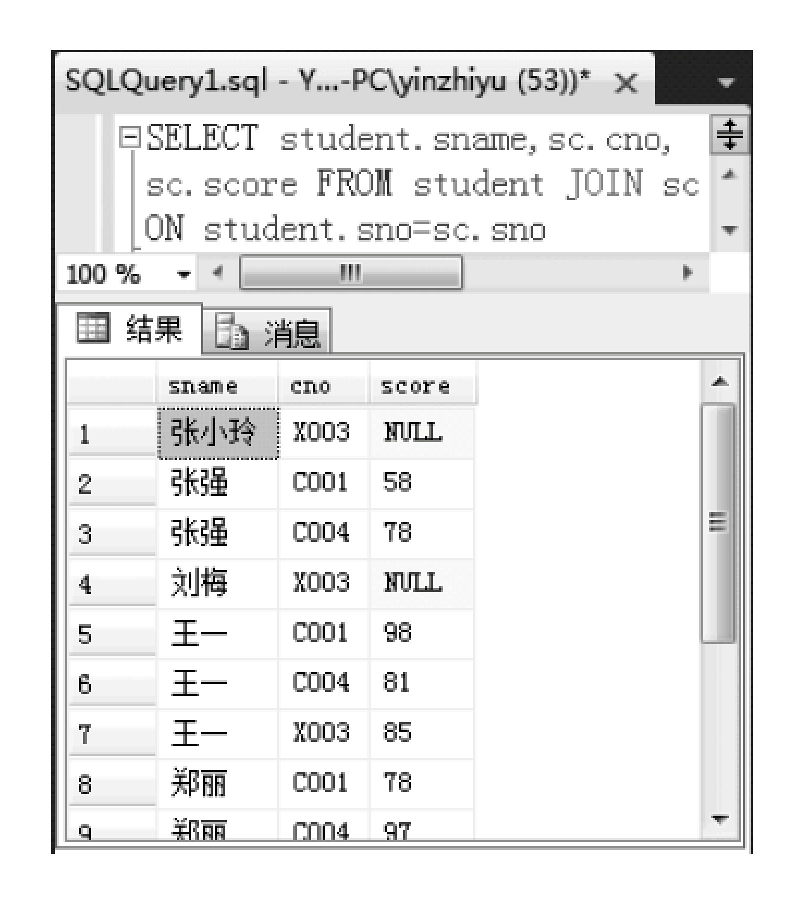


图 4-18　例 4-26 的查询结果

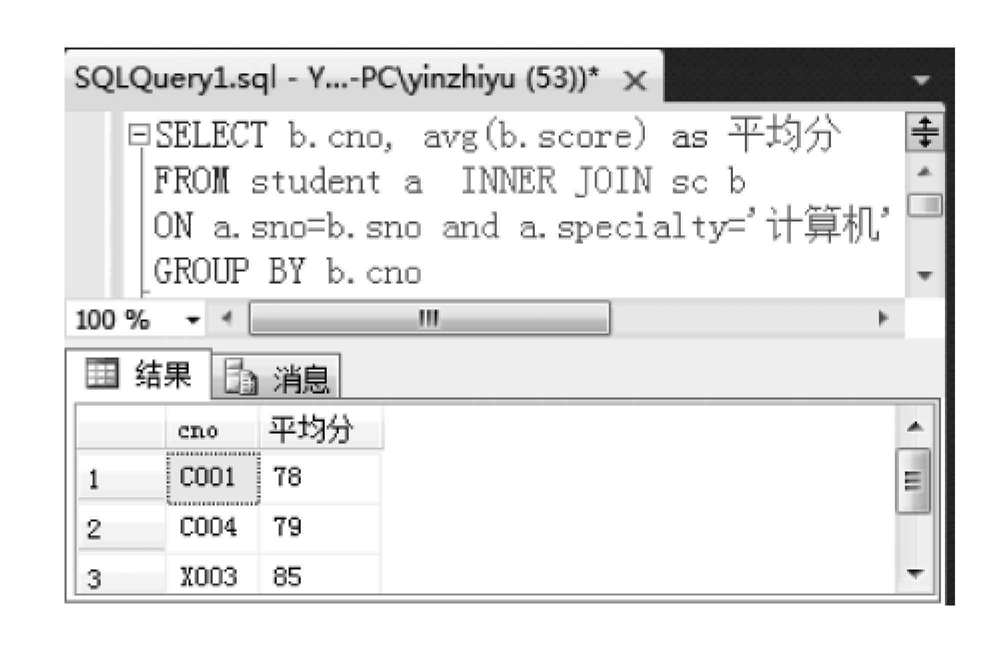


图 4-19　例 4-27 的查询结果

为了简化输入，可以在 SELECT 查询的 FROM 子句中为表定义一个临时别名，在查询中引用，以缩写表名。

4.4.2　自连接

连接操作不仅可以在不同的表上进行，而且在同一张表内可以进行自身连接，即将同一个表的不同行连接起来。自连接可以看作一张表的两个副本之间的连接。在自连接中，必须为表指定两个别名，使之在逻辑上成为两张表。

【例 4-28】　从 teaching 数据库中查询同名学生的信息。

```
USE teaching
SELECT * FROM student a   INNER JOIN student b
ON a.sname = b.sname AND a.sno <> b.sno
```

查询结果如图 4-20 所示。

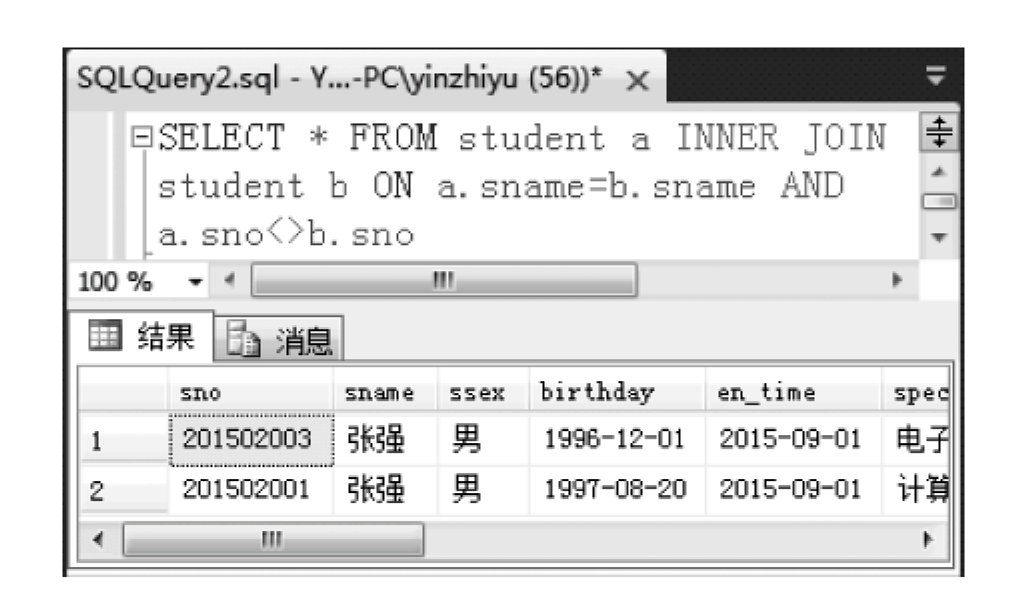


图 4-20　例 4-28 查询结果

4.4.3　外连接

在外连接中，不仅包括那些满足条件的数据，而且某些表不满足条件的数据也会显示在结果集中。也就是说，外连接一般只限制其中一个表的数据行，而不限制另外一个表中的数据。这种连接形式在许多情况下是非常有用的，例如在连锁超市统计报表时，不仅要统计那些有销售量的超市和商品，而且还要统计那些没有销售量的超市和商品。



1）外连接的分类

在 Microsoft SQL Server 2008 系统中，可以使用的 3 种外连接关键字，即 LEFT OUTER JOIN、RIGHT OUTER JOIN 和 FULL OUTER JOIN。

① 左外连接是对连接条件中左边的表不加限制；

② 右外连接是对右边的表不加限制；

③ 全外连接对两个表都不加限制，所有两个表中的行都会包括在结果集中。

2）外连接的语法

（1）左外连接。

```
SELECT select_list
FROM  表 1  LEFT [OUTER] JOIN  表 2
ON 表 1.列 1 = 表 2.列 2
```

（2）右外连接。

```
SELECT select_list
FROM  表 1 RIGHT[OUTER]JOIN 表 2
ON 表 1.列 1 = 表 2.列 2
```

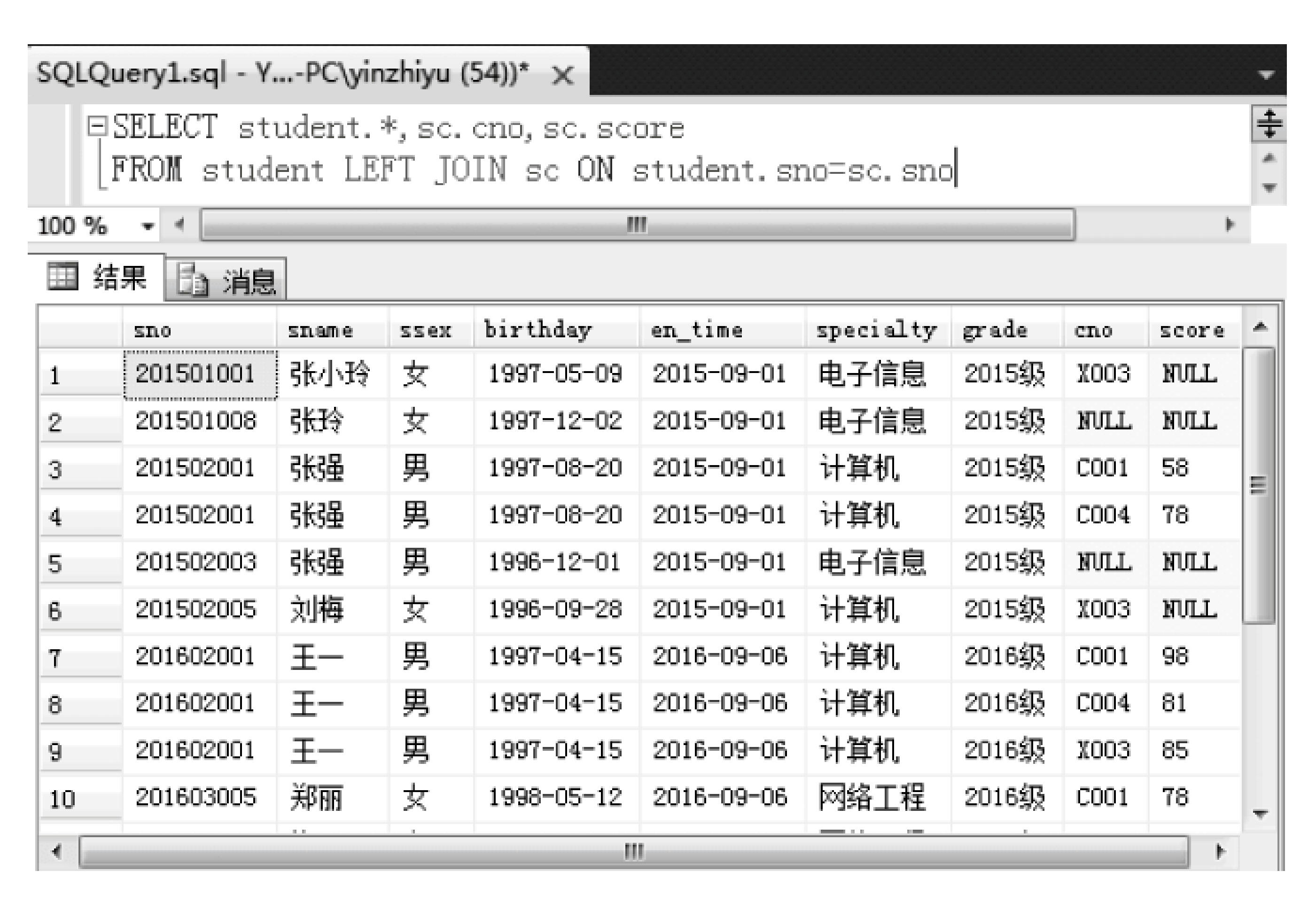
（3）全外连接。

```
SELECT select_list
FROM  表 1 FULL[OUTER]  JOIN  表 2
ON 表 1.列 1 = 表 2.列 2
```

【例 4-29】 在 teaching 数据库中查询每个学生及其选修课程的成绩情况（含未选课的学生信息）。

```
USE teaching
SELECT student.*,sc.cno,sc.score
FROM student LEFT JOIN sc
ON student.sno = sc.sno
```

查询结果如图 4-21 所示。



SQLQuery1.sql - Y...-PC\yinzhiyu (54))*

```
SELECT student.*, sc.cno, sc.score
FROM student LEFT JOIN sc ON student.sno=sc.sno
```

100 %

结果 消息

	sno	sname	ssex	birthday	en_time	specialty	grade	cno	score
1	201501001	张小玲	女	1997-05-09	2015-09-01	电子信息	2015级	X003	NULL
2	201501008	张玲	女	1997-12-02	2015-09-01	电子信息	2015级	NULL	NULL
3	201502001	张强	男	1997-08-20	2015-09-01	计算机	2015级	C001	58
4	201502001	张强	男	1997-08-20	2015-09-01	计算机	2015级	C004	78
5	201502003	张强	男	1996-12-01	2015-09-01	电子信息	2015级	NULL	NULL
6	201502005	刘梅	女	1996-09-28	2015-09-01	计算机	2015级	X003	NULL
7	201602001	王一	男	1997-04-15	2016-09-06	计算机	2016级	C001	98
8	201602001	王一	男	1997-04-15	2016-09-06	计算机	2016级	C004	81
9	201602001	王一	男	1997-04-15	2016-09-06	计算机	2016级	X003	85
10	201603005	郑丽	女	1998-05-12	2016-09-06	网络工程	2016级	C001	78

图 4-21　例 4-29 的查询结果

【例 4-30】 在 teaching 数据库中查询每个学生及其选修课程的情况(含未选课的学生信息及未被选修的课程信息)。

```
USE teaching
SELECT course. * , sc. score, student. sname, student. sno
FROM course FULL JOIN sc ON course. cno = sc. cno
FULL JOIN student ON student. sno = sc. sno
```

查询结果如图 4-22 所示。

SQLQuery2.sql - Y...-PC\yinzhiyu (56))*

```
SELECT course.*, sc.score, student.sname, student.sno
FROM course FULL JOIN sc ON course.cno=sc.cno
FULL JOIN student ON student.sno=sc.sno
```

结果 消息

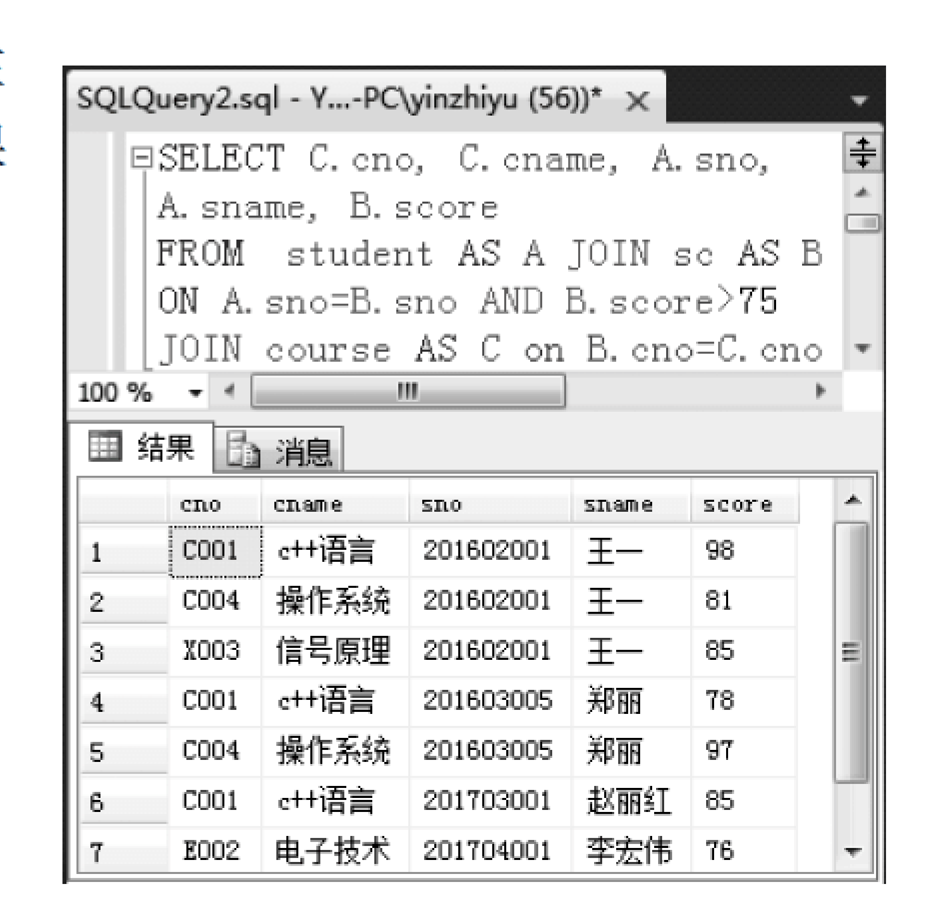
	cno	cname	classhour	credit	score	sname	sno
1	C001	c++语言	4	4	98	王一	201602001
2	C001	c++语言	4	4	78	郑丽	201603005
3	C001	c++语言	4	4	72	沈艳	201701003
4	C001	c++语言	4	4	85	赵丽红	201703001
5	C004	操作系统	3	3	81	王一	201602001
6	C004	操作系统	3	3	97	郑丽	201603005
7	E002	电子技术	5	5	73	沈艳	201701003
8	E002	电子技术	5	5	76	李宏伟	201704001
9	R005	软件工程	3	3	NULL	NULL	NULL
10	X003	信号原理	4	3	NULL	张小玲	201501001
11	X003	信号原理	4	3	NULL	刘梅	201502005
12	X003	信号原理	4	3	85	王一	201602001
13	NULL	NULL	NULL	NULL	NULL	张强	201502003
14	NULL	NULL	NULL	NULL	NULL	刘景鹏	201704006
15	NULL	NULL	NULL	NULL	NULL	张强	201502001
16	NULL	NULL	NULL	NULL	NULL	张玲	201501008

图 4-22 例 4-30 的查询结果

【例 4-31】 在 teaching 数据库中查询成绩在 75 分以上的学生的学号、姓名,选修课的课程号、课程名、成绩。

```
 USE teaching
SELECT C. cno, C. cname, A. sno, A. sname, B. score
FROM   student AS A JOIN sc AS B
ON A. sno = B. sno AND B. score > 75
JOIN course AS C on B. cno = C. cno
```

查询结果如图 4-23 所示。


SQLQuery2.sql - Y...-PC\yinzhiyu (56))*

```
SELECT C.cno, C.cname, A.sno,
A.sname, B.score
FROM student AS A JOIN sc AS B
ON A.sno=B.sno AND B.score>75
JOIN course AS C on B.cno=C.cno
```

结果 消息

	cno	cname	sno	sname	score
1	C001	c++语言	201602001	王一	98
2	C004	操作系统	201602001	王一	81
3	X003	信号原理	201602001	王一	85
4	C001	c++语言	201603005	郑丽	78
5	C004	操作系统	201603005	郑丽	97
6	C001	c++语言	201703001	赵丽红	85
7	E002	电子技术	201704001	李宏伟	76

图 4-23 例 4-31 的查询结果

4.4.4 交叉连接

交叉连接也被称为笛卡儿乘积,返回两个表的乘积。在检索结果集中,包含了所连接的两个表中所有行的全部组合。

例如,如果对 A 表和 B 表执行交叉连接,A 表中有 5 行数据,B 表中有 12 行数据,那么结果集中可以有 60 行数据。

交叉连接使用 CROSS JOIN 关键字来创建。实际上,交叉连接的使用是很少的,但是交叉连接是理解外连接和内连接的基础。语法格式如下:

```
SELECT 列
FROM 表 1 CROSS JOIN 表 2
```

【例 4-32】 在 teaching 数据库中查询所有学生可能的选课情况。

```
USE teaching
SELECT a. * ,b.cno,b.score
FROM student a CROSS JOIN sc b
```

查询结果如图 4-24 所示。



	sno	sname	ssex	birthday	en_time	specialty	grade	cno	score
18	201603005	郑丽	女	1998-05-12	2016-09-06	网络工程	2016级	X003	NULL
19	201701003	沈艳	女	1999-09-01	2017-09-05	电子信息	2017级	X003	NULL
20	201703001	赵丽红	女	1999-08-02	2017-09-05	网络工程	2017级	X003	NULL
21	201704001	李宏伟	男	1998-10-16	2017-09-05	通信工程	2017级	X003	NULL
22	201704006	刘景鹏	男	1998-02-09	2017-09-05	通信工程	2017级	X003	NULL
23	201501001	张小玲	女	1997-05-09	2015-09-01	电子信息	2015级	C001	98
24	201501008	张玲	女	1997-12-02	2015-09-01	电子信息	2015级	C001	98
25	201502001	张强	男	1997-08-20	2015-09-01	计算机	2015级	C001	98
26	201502003	张强	男	1996-12-01	2015-09-01	电子信息	2015级	C001	98
27	201502005	刘梅	女	1996-09-28	2015-09-01	计算机	2015级	C001	98
28	201602001	王一	男	1997-04-15	2016-09-06	计算机	2016级	C001	98

图 4-24 例 4-32 的查询结果

4.5 子 查 询

SELECT 语句可以嵌套在其他许多语句中,这些语句包括 SELECT、INSERT、UPDATE 及 DELETE 等,这些嵌套的 SELECT 语句被称为子查询。

当一个查询依赖于另外一个查询结果时,那么可以使用子查询(一般为查询条件未知)。在某些查询中,查询语句比较复杂、不容易理解,为了把这些复杂的查询语句分解成多个比较简单的查询语句形式,也常使用子查询方式。

使用子查询方式完成查询操作的技术是子查询技术。子查询可以分为无关子查询(嵌套子查询)和相关子查询。

4.5.1 无关子查询

无关子查询的执行不依赖于外部查询。无关子查询在外部查询之前执行,然后返回数据供外部查询使用,无关子查询中不包含对于外部查询的任何引用。

1. 比较子查询

使用子查询进行比较测试时,通过等于(=)、不等于(<>)、小于(<)、大于(>)、小于

或等于(<=)以及大于或等于(>=)等比较运算符,将一个表达式的值与子查询返回的单值进行比较。如果比较运算的结果为 TRUE,则比较测试也返回 TRUE。

【例 4-33】 在 teaching 数据库中查询与“沈艳”在同一个专业学习的学生的学号、姓名和专业。

```
USE teaching
SELECT sno, sname, specialty
FROM student
WHERE specialty =
(SELECT specialty FROM  student
 WHERE sname = '沈艳')
```

查询结果如图 4-25 所示。

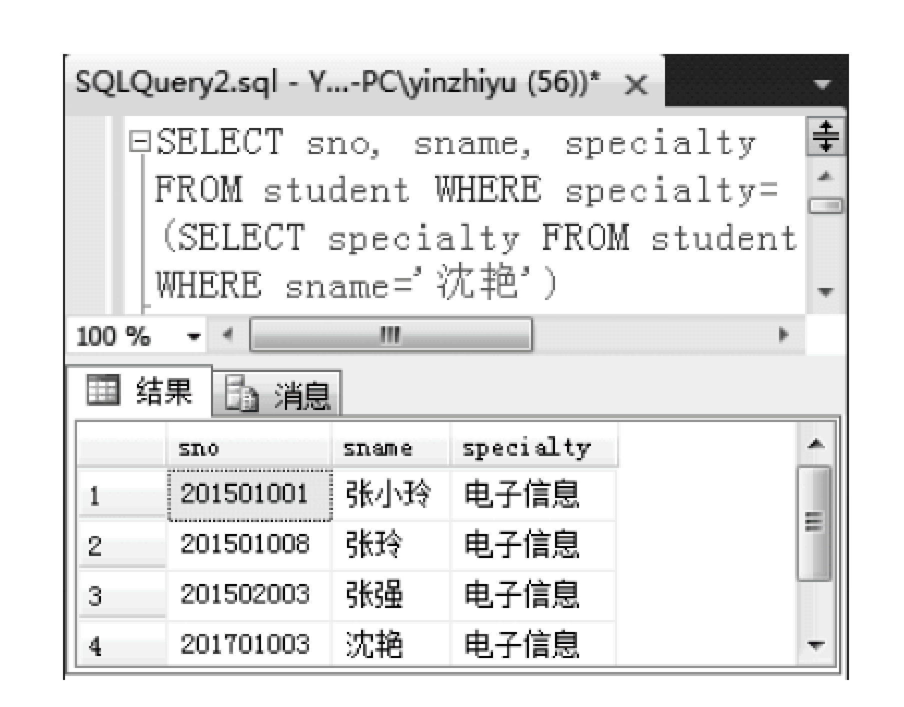


图 4-25 例 4-33 的查询结果

例 4-33 可以用自连接来实现,程序如下:

```
USE teaching
SELECT a.sno, a.sname, a.specialty
FROM student a, student b
WHERE a. specialty = b. specialty AND b. sname = '沈艳'
```

需要特别指出的是,子查询的 SELECT 语句不能使用 ORDER BY 子句,ORDER BY 子句只能对最终查询结果排序。

【例 4-34】 在 teaching 数据库中查询 C001 号课的考试成绩比“郑丽”高的学生的学号和姓名。

```
USE teaching
SELECT student.sno,sname FROM student,sc
WHERE student.sno  =  sc.sno and cno = 'C001'
and score >(SELECT score FROM sc WHERE cno = 'C001' and
           sno = (SELECT sno FROM student WHERE sname = '郑丽'))
```

查询结果如图 4-26 所示。

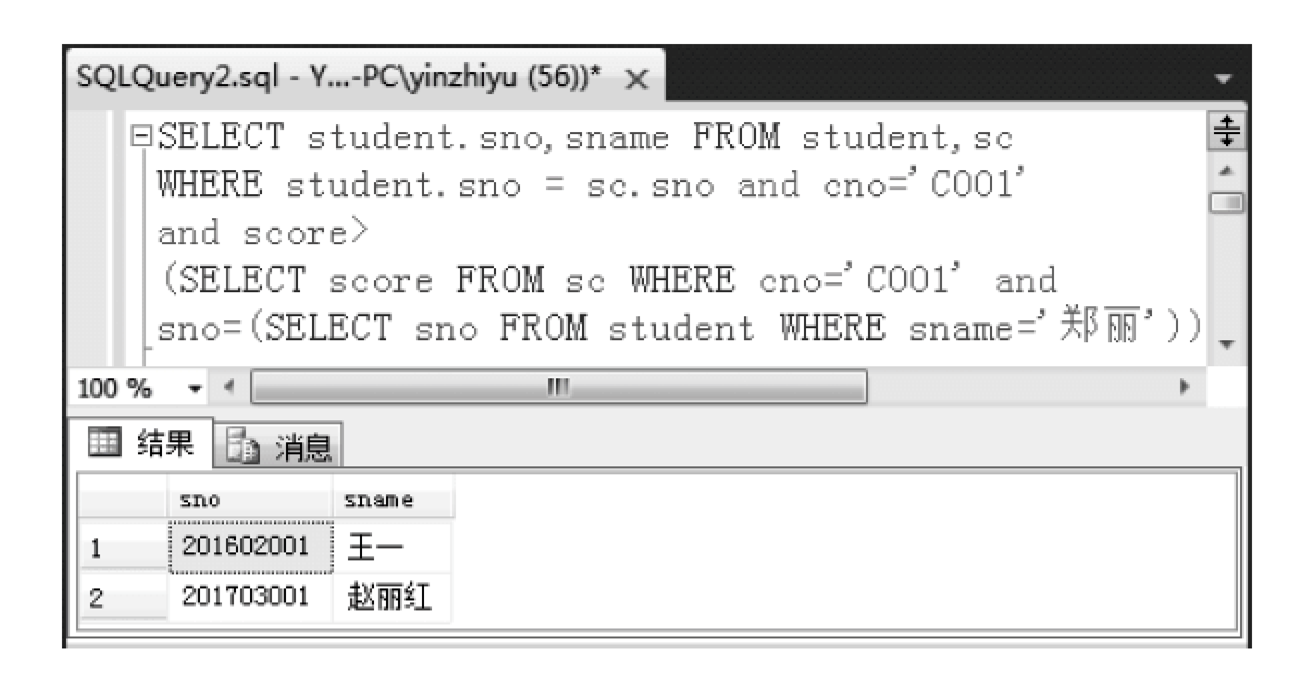


图 4-26 例 4-34 的查询结果

2. SOME、ANY、ALL 和 IN 子查询

ALL 和 ANY 操作符的常见用法是结合一个比较操作符对一个数据列子查询的结果进行测试。它们测试比较值是否与子查询所返回的全部或一部分值匹配。比如说,如果比较值小于或等于子查询所返回的每一个值,<= ALL 将是 true;只要比较值小于或等于子

查询所返回的任何一个值，<= ANY 就将是 true。SOME 是 ANY 的一个同义词。

【例 4-35】 查询 teaching 数据库中计算机专业年龄最大的学生的学号和姓名。

```
USE teaching
SELECT sno, sname FROM student WHERE sbirthday <= ALL
(SELECT sbirthday FROM student WHERE specialty = '计算机')
  AND specialty = '计算机'
```

查询结果如图 4-27 所示。

【例 4-36】 查询 teaching 数据库中与任何一个通信工程专业同龄的学生的信息。

```
USE teaching
SELECT * FROM student WHERE year(birthday) = ANY
(SELECT year(birthday) FROM student WHERE specialty = '通信工程')
```

查询结果如图 4-28 所示。

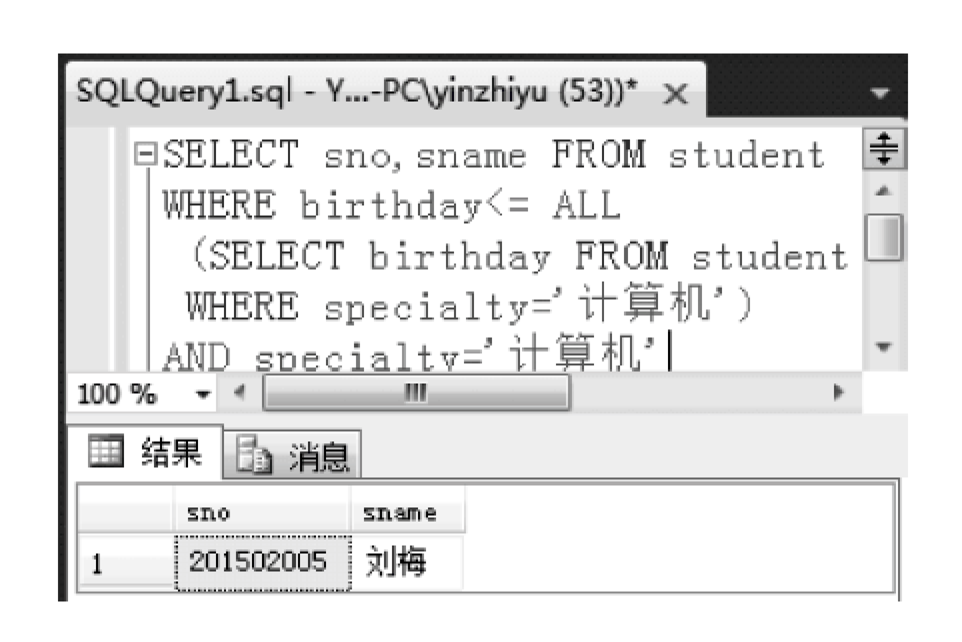


图 4-27　例 4-35 的查询结果

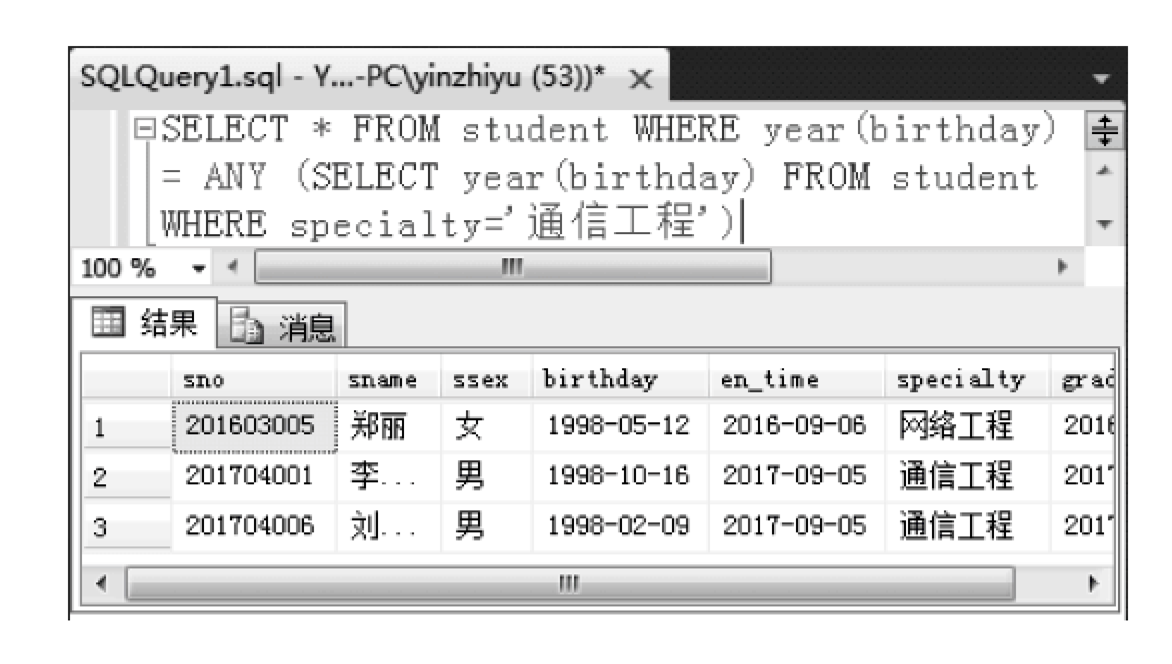


图 4-28　例 4-36 的查询结果

实际上，IN 和 NOT IN 操作符是= ANY 和< > ALL 的简写。也就是说，IN 操作符的含义是“等于子查询所返回的某个数据行”，NOT IN 操作符的含义是“不等于子查询所返回的任何数据行”。

【例 4-37】 在 teaching 数据库中查询选修了 C001 号课程的学生姓名和所在专业。

```
USE teaching
SELECT sname, specialty FROM student
WHERE sno IN
    (SELECT sno FROM sc WHERE cno = 'C001')
```

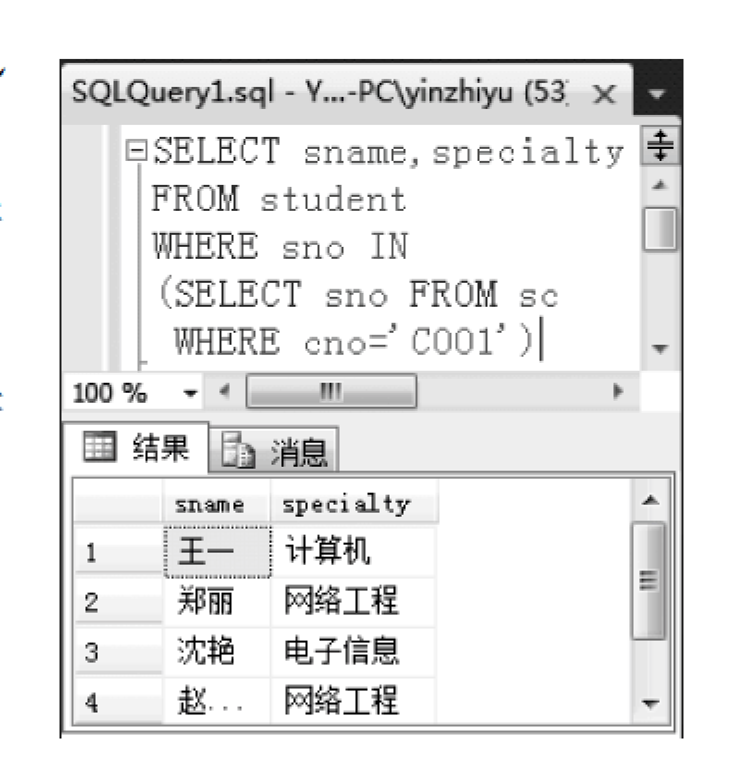


图 4-29　例 4-37 的查询结果

查询结果如图 4-29 所示。

3. 子查询结果作为主查询的查询对象

【例 4-38】 在 teaching 数据库中查询有 2 个以上学生平均成绩超过 80 分的班级(用年级和专业表示)。

```
USE teaching
SELECT grade, specialty FROM student s,
(SELECT sno FROM SC GROUP BY sno HAVING AVG(score) >= 80) ss
```

```
        WHERE s.sno = ss.sno
GROUP BY grade,specialty
 HAVING COUNT( * )>= 2
```

4.5.2 相关子查询

在相关子查询中，子查询的执行依赖于外部查询，多数情况下是子查询的 WHERE 子句中引用了外部查询的表。

相关子查询的执行过程与无关子查询完全不同，无关子查询中子查询只执行一次，而相关子查询中的子查询需要重复执行。

相关子查询的执行过程如下：

① 子查询为外部查询的每一行执行一次，外部查询将子查询引用的列的值传给子查询。

② 如果子查询的任何行与其匹配，外部查询就返回结果行。

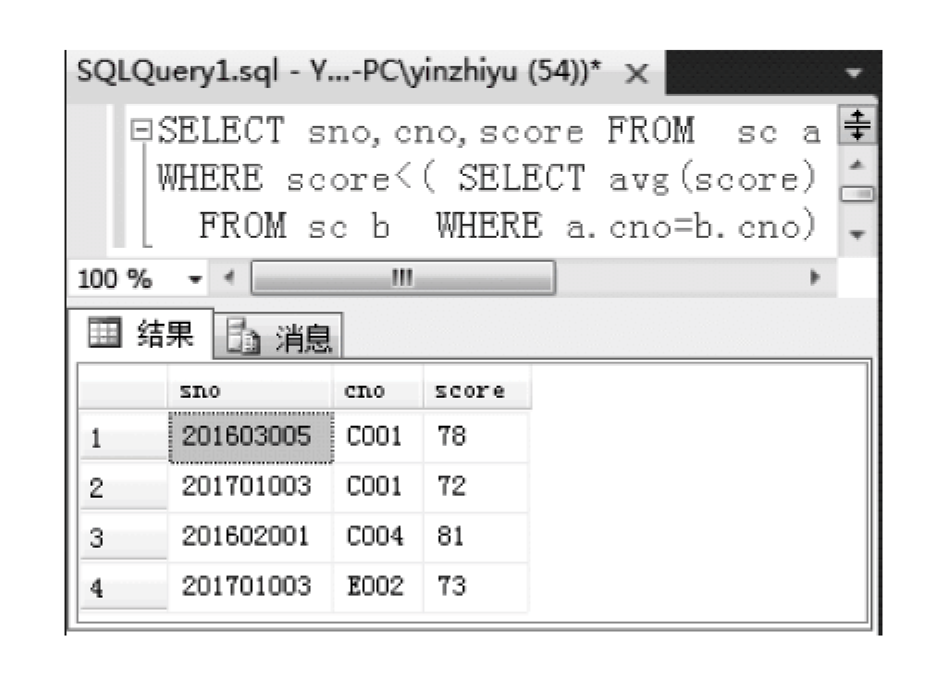
③ 再回到第一步，直到处理完外部表的每一行。

1. 比较子查询

【例 4-39】 在 teaching 数据库中查询成绩比该课的平均成绩低的学生的学号、课程号、成绩。

```
USE teaching
SELECT sno,cno,score FROM  sc a  WHERE score <( SELECT avg(score)  FROM sc b  WHERE a.cno =
b.cno)
```

查询结果如图 4-30 所示。



	sno	cno	score
1	201603005	C001	78
2	201701003	C001	72
3	201602001	C004	81
4	201701003	E002	73

图 4-30　例 4-39 的查询结果

【例 4-40】 在 teaching 数据库中查询有 2 门以上课程的成绩在 80 分以上的学生的学号、姓名、年级和专业。

```
SELECT sno,sname,grade,specialty FROM student s
WHERE (SELECT COUNT( * ) FROM sc
        WHERE sc.sno = s.sno and score >= 80)>= 2
```

2. 带有 EXISTS 的子查询(存在性测试)

使用子查询进行存在性测试时，通过逻辑运算符 EXISTS 或 NOT EXISTS，检查子查询所返回的结果集是否有行存在。使用逻辑运算符 EXISTS 时，如果在子查询的结果集内

包含有一行或多行，则存在性测试返回 TRUE；如果该结果集内不包含任何行，则存在性测试返回 FALSE。在 EXISTS 前面加上 NOT 时，将对存在性测试结果取反。

带有 EXISTS 谓词的子查询不返回任何数据，只产生逻辑真值 TRUE 或逻辑假值 FALSE。

【例 4-41】 在 teaching 数据库中查询所有选修了 C004 号课程的学生姓名。

分析：本查询涉及 student 表和 sc 表。可以在 student 表中依次取每个元组的 sno 值，用此值去检查 sc 表。若 sc 表中存在这样的元组，其 sno 值等于 student 表中 sno 的值，并且 cno='C004'，则取此学生的姓名送入结果关系。当然，此查询是完全可以用无关子查询来完成的，请读者自行完成，并试着分析比较二者的查询效率。

```
USE teaching
SELECT sname FROM student
WHERE EXISTS
  (SELECT * FROM sc
   WHERE sno = student.sno
AND cno = 'C004')
```

查询结果如图 4-31 所示。

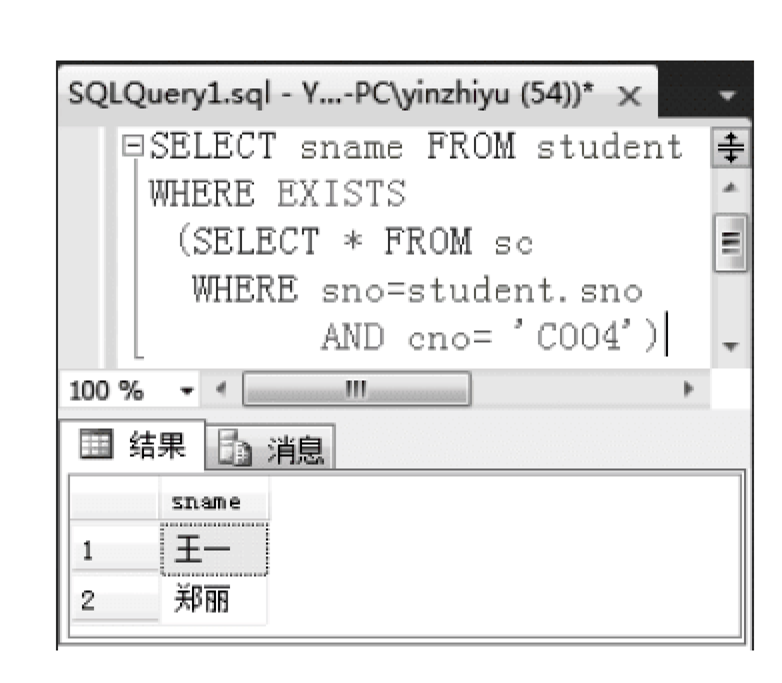


图 4-31 例 4-41 的查询结果

由 EXISTS 引出的子查询，其目标属性列表达式一般用 * 表示，因为带 EXISTS 的子查询只返回真值或假值，给出列名无实际意义。

若内层子查询结果非空，则外层的 WHERE 子句条件为真(TRUE)，否则为假(FALSE)。

使用子查询时要注意以下几点：

(1) 子查询需要用小括号( )括起来。

(2) 子查询可以嵌套。

(3) 子查询的 SELECT 语句中不能使用 image、text 和 ntext 数据类型。

(4) 子查询返回的结果的数据类型必须匹配外围查询 WHERE 语句的数据类型。

(5) 子查询中不能使用 ORDER BY 子句。

4.6 其他查询

4.6.1 集合运算查询

1. UNION 联合查询

联合查询是指将两个或两个以上的 SELECT 语句通过 UNION 运算符连接起来的查询，联合查询可以将两个或更多查询的结果组合为单个结果集，该结果集包含联合查询中所有查询的全部行。

使用 UNION 组合两个查询结果集的两个基本规则是：所有查询中的列数和列的顺序必须相同；数据类型必须兼容。

其语法格式如下：

```
Select_statement
UNION [ ALL ]  Select_statement
[ UNION [ ALL ]  Select_statement [ …n ] ]
```

其中的参数说明如下：

① Select_statement——是参与查询的 SELECT 语句。

② ALL——在结果中包含所有的行，包括重复行；如果没有指定，则删除重复行。

【例 4-42】 查询选修了课程 C001 和课程 C004 的学生的姓名和课程号。

```
USE teaching
SELECT cno, sname FROM sc, student
WHERE cno = 'C001' and sc. sno = student. sno
UNION
SELECT cno, sname FROM sc, student
WHERE cno = 'C004' and sc.  sno = student. sno
```

查询结果如图 4-32 所示。

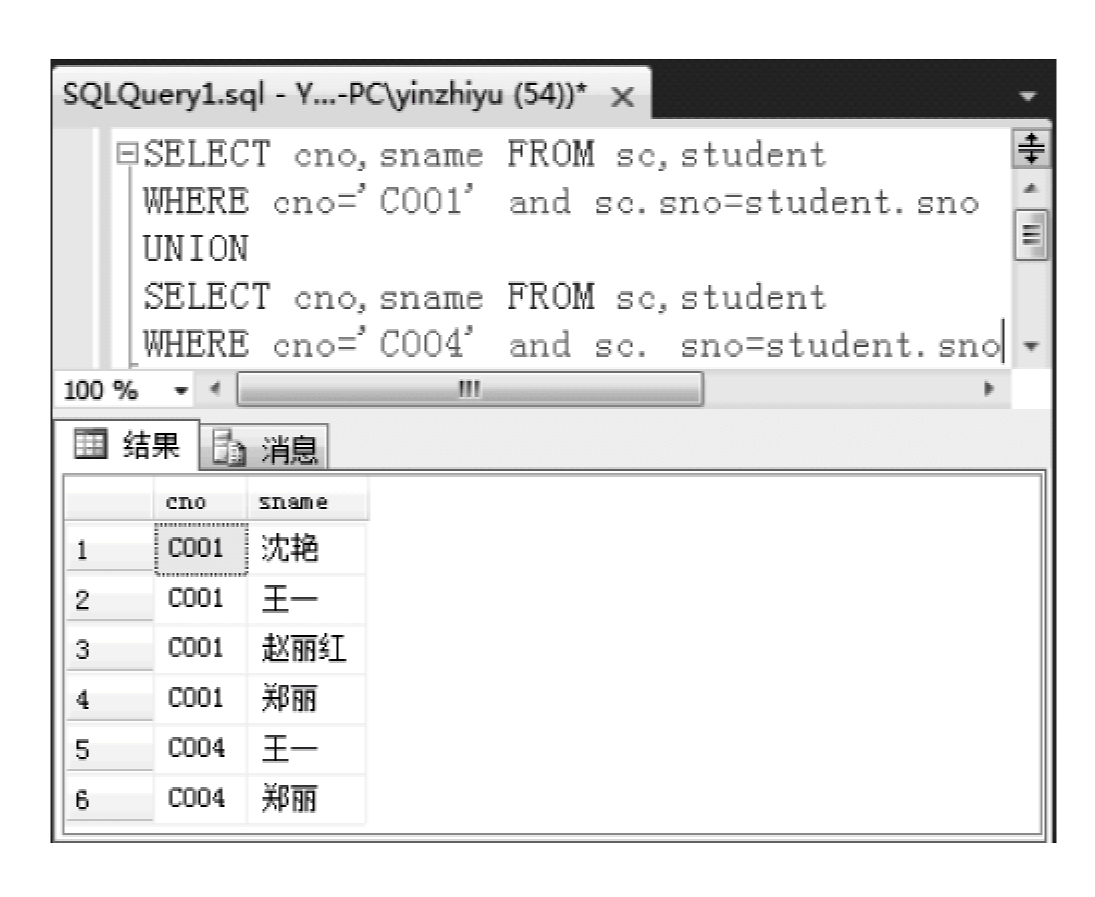


图 4-32 例 4-42 的查询结果

2. EXCEPT 和 INTERSECT 查询

EXCEPT 和 INTERSECT 运算符可以比较两个或多个 SELECT 语句的结果并返回非重复值。EXCEPT 运算符返回由 EXCEPT 运算符左侧的查询返回、而又不包含在右侧查询所返回的值中的所有非重复值。INTERSECT 返回由 INTERSECT 运算符左侧和右侧的查询都返回的所有非重复值。使用 EXCEPT 和 INTERSECT 的基本规则同 UNION。

语法格式如下：

```
Select_statement
{EXCEPT | INTERSECT}
Select_statement
```

【例 4-43】 查询计算机专业没有选修操作系统课程的学生的学号和姓名。

```
SELECT sno, sname FROM student WHERE specialty = '计算机'
EXCEPT
SELECT sc. sno, sname FROM sc, student WHERE sc. sno = student. sno AND specialty = '计算机' AND cno
IN (SELECT cno FROM course WHERE cname  = '操作系统')
```

执行结果如图 4-33 所示。

【例 4-44】 查询既选修了 C001 又选修了 C004 号课程的学生的学号。

```
SELECT sno FROM sc WHERE cno = 'C001'
INTERSECT
SELECT sno FROM sc WHERE cno = 'C004'
```

执行结果如图 4-34 所示。

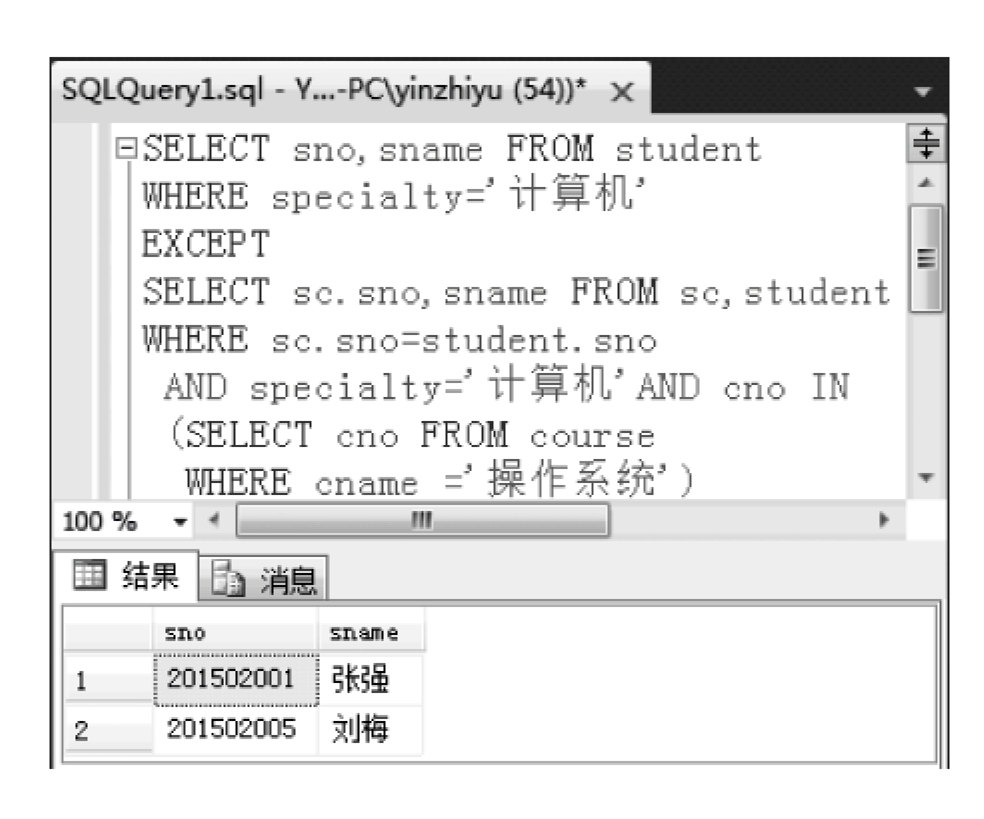


图 4-33 例 4-43 的查询结果

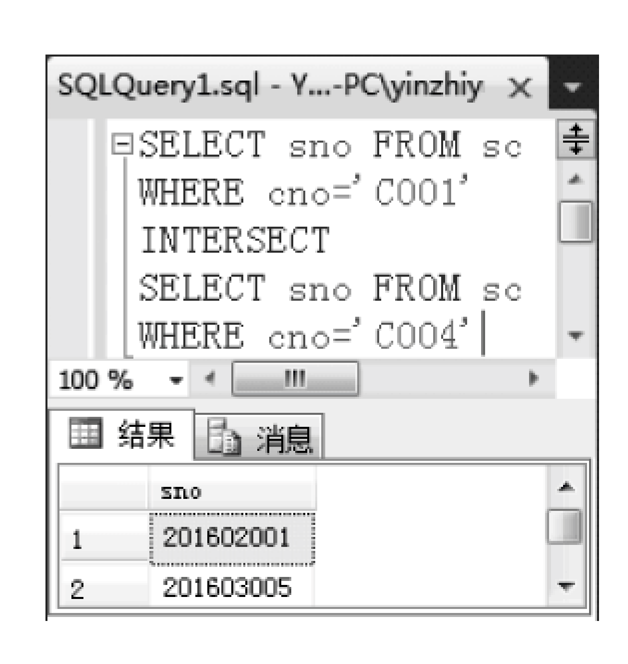


图 4-34 例 4-44 的查询结果

4.6.2 对查询结果排序

在使用 SELECT 语句时，排序是一种常见的操作。

排序是指按照指定的列或其他表达式对结果集排列顺序的方式。SELECT 语句中的 ORDER BY 子句负责完成排序操作。

其语法格式如下：

```
[ ORDER BY { order_by_expression [ ASC | DESC ] } [ ,…n ] ]
```

其中的参数说明如下：

① order_by_expression——指定要排序的列。可以指定多个列。在 ORDER BY 子句中不能使用 ntext、text 和 image 列。

② ASC 表示升序，DESC 表示降序，默认情况下是升序。

【例 4-45】 查询 teaching 数据库中“女”学生的姓名和专业，并按姓名升序排列。

```
USE teaching
SELECT sname, specialty FROM student
WHERE ssex = '女'
ORDER BY sname ASC
```

查询结果如图 4-35 所示。

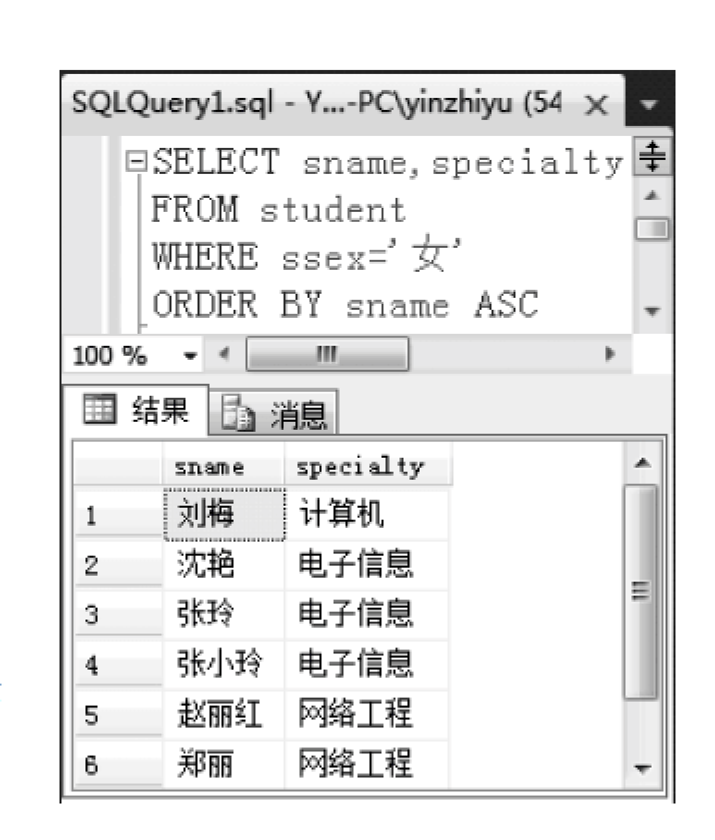


图 4-35 例 4-45 的查询结果

【例 4-46】 查询 sc 表中学生的成绩和学号，并按成绩降序排列。

```
USE teaching
```

```
SELECT sno,score FROM sc
ORDER BY score DESC
```

【例 4-47】 查询 sc 中学生的成绩和学号,并按成绩(降序)排列,若成绩相同,则按学号(升序)排列。

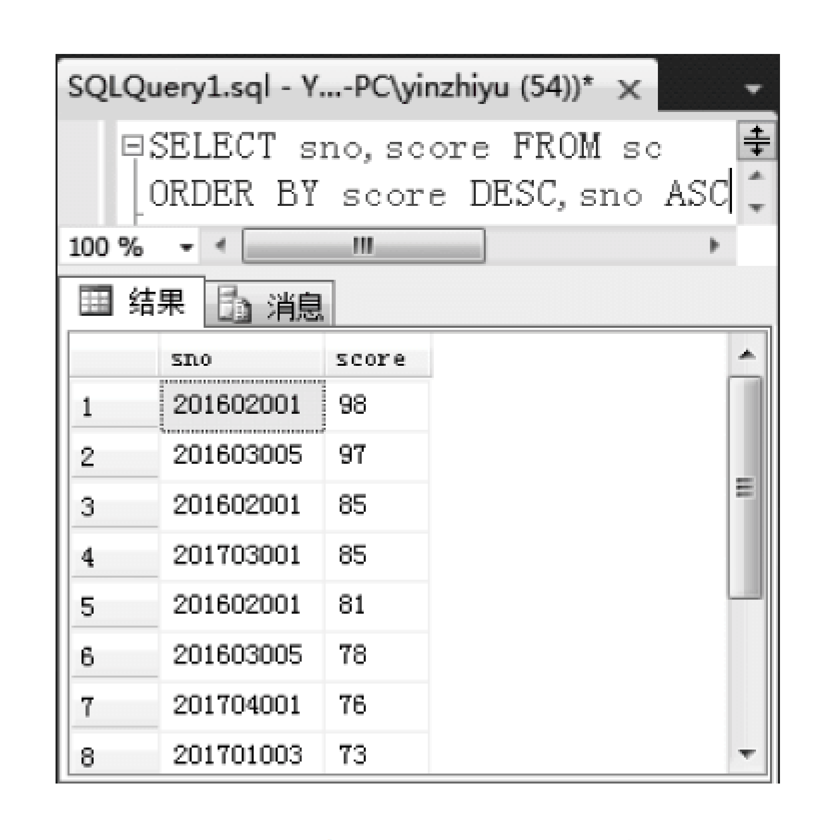
```
USE  teaching
SELECT sno,score FROM sc
ORDER BY score DESC,sno ASC
```

查询结果如图 4-36 所示。

【例 4-48】 使用 TOP 关键字查询平均成绩最高的前三名。

```
USE  teaching
SELECT TOP 3 sno,AVG(score) 平均成绩 FROM sc
GROUP BY sno
ORDER BY AVG(score) DESC
```

查询结果如图 4-37 所示。



	sno	score
1	201602001	98
2	201603005	97
3	201602001	85
4	201703001	85
5	201602001	81
6	201603005	78
7	201704001	76
8	201701003	73

图 4-36 例 4-47 的查询结果



	sno	平均成绩
1	1302001	88
2	1302005	87
3	1402001	85

图 4-37 例 4-48 的查询结果

4.6.3 存储查询结果

通过在 SELECT 语句中使用 INTO 子句,可以创建一个新表并将查询结果中的行添加到该表中。用户在执行一个带有 INTO 子句的 SELECT 语句时,必须拥有在目标数据库上创建表的权限。

SELECT...INTO 语句的语法格式如下:

```
SELECT select_list INTO new_table
FROM table_source
[WHERE search_condition]
```

其中,new_table 为要新建的表的名称。新表中包含的列由 SELECT 子句中选择列表的内容决定,新表中包含的行数则由 WHERE 子句指定的搜索条件决定。

【例 4-49】 在 teaching 数据库中将查询的学生姓名、学号、课程名、成绩的相关数据存放在表"成绩单"中,并对新表进行查询。

```
USE teaching
SELECT sname,student.sno, cname, score INTO 成绩单
FROM student,sc,course
WHERE student.sno = sc.sno AND course.cno = sc.cno
GO
SELECT * FROM 成绩单
```

查询结果如图 4-38 所示。



图 4-38　例 4-49 的查询结果

4.7　在数据操作中使用 SELECT 子句

可以在 INSERT 语句、UPDATE 语句和 DELETE 语句中使用 SELECT 子句(子查询),以完成相应的数据插入、修改和删除操作。

4.7.1　在 INSERT 语句中使用 SELECT 子句

在 INSERT 语句中使用 SELECT 子句可以将一个或多个表或视图中的值添加到另一个表中。使用 SELECT 子句还可以同时插入多行。

在 INSERT 语句中使用 SELECT 子句的语法形式为:

```
INSERT [INTO] table_name[(column_list)]
SELECT select_list
FROM table_name
[WHERE search_condition]
```

【例 4-50】 在 teaching 数据库中创建 sc 表的一个副本——“成绩表”,将 sc 中成绩大于 80 的数据添加到“成绩表”中,并显示表中内容。

```
USE teaching
CREATE TABLE 成绩表
```

```
        (学号   char(7),
         课程号 char(4) ,
         成绩   int )
GO
INSERT INTO 成绩表(学号,课程号,成绩)
SELECT * FROM  sc
WHERE score>=80
GO
SELECT * FROM 成绩表
```

注意：

① 不要把 SELECT 子句写在小括号中。

② INSERT 语句中的列名列表应当放在小括号中，而且不使用 VALUES 关键字。如果来源表与目标表结构完全相同，则可以省略 INSERT 语句中的列名列表。

③ SELECT 子句中的列列表必须与 INSERT 语句中的列列表相匹配。如果没有在 INSERT 语句中给出列列表，那么 SELECT 子句中的列列表必须与目标表中的列相匹配。

4.7.2 在 UPDATE 语句中使用 SELECT 子句

在 UPDATE 语句中使用 SELECT 子句可以将子查询的结果作为修改数据的条件。

在 UPDATE 语句中使用 SELECT 子句的语法形式为：

```
UPDATE table_name
SET { column_name = { expression } } [,...n ]
  [WHERE {condition_expression}]
```

其中，condition_expression 中包含 SELECT 子句，SELECT 子句要写在小括号中。

【例 4-51】 在 teaching 数据库中将 201602001 号学生选修的"操作系统"课的成绩改为 86 分。

```
UPDATE sc SET score=86 WHERE sno='201602001' AND
   cno=(SELECT cno FROM course WHERE cname='操作系统')
```

【例 4-52】 在 teaching 数据库中将 2016 级计算机专业王一选修的 C001 号课的成绩改为 92 分。

方法一：使用 SELECT 子句。

```
UPDATE sc SET score=92 WHERE cno='C001' AND
  sno=(SELECT sno FROM student WHERE sname='王一'
 AND grade='2016 级' AND specialty='计算机')
```

方法二：使用 JOIN 内连接。

```
UPDATE sc SET score=92 FROM sc JOIN student ON
  student.sno=sc.sno WHERE cno='C001' AND
  sname='王一' AND grade='2016 级' AND specialty='计算机'
```

4.7.3 在 DELETE 语句中使用 SELECT 子句

在 DELETE 语句中使用 SELECT 子句可以将子查询的结果作为删除数据的条件。

在 DELETE 语句中使用 SELECT 子句的语法形式为：

```
DELETE  [FROM]  table_name
  [WHERE {condition_expression}]
```

其中，condition_expression 中包含 SELECT 子句，SELECT 子句要写在小括号中。

【例 4-53】 在 teaching 数据库中将 201602001 号学生选修的“操作系统”课删除。

```
DELETE sc WHERE sno = '201602001' AND
  cno = (SELECT cno FROM course WHERE cname = '操作系统')
```

【例 4-54】 在 teaching 数据库中将 2016 级计算机专业王一选修的 C001 号课删除。

方法一：使用 SELECT 子句。

```
DELETE sc WHERE cno = 'C001' AND sno =
(SELECT sno FROM student WHERE sname = '王一'
AND grade = '2016 级' AND specialty = '计算机')
```

方法二：使用 JOIN 内连接。

```
DELETE sc FROM sc JOIN student ON
student.sno = sc.sno WHERE cno = 'C001' AND
sname = '王一'  AND grade = '2016 级' AND specialty = '计算机'
```

习　题　4

1. 针对 teaching 数据库中的三个表，试用 T-SQL 的查询语句实现下列查询。

(1) 查询学生们有哪些专业，只显示专业列，过滤掉重复行。

(2) 统计有学生选修的课程门数。

(3) 求 C004 号课程的平均成绩。

(4) 求学分为 3 的各门课程的平均成绩。

(5) 统计每门课程的选修人数，超过两人的课程才统计。要求输出课程号和选修人数，查询结果按人数降序排列；若人数相同，则按课程号升序排列。

(6) 查询所有姓刘的学生的姓名和学号。

(7) 查询成绩为空值的学生学号和课程号。

(8) 查询没有学生选修的课程的课程号和课程名。

(9) 求年龄大于男同学平均年龄的女学生的姓名和学号。

(10) 求年龄大于所有男同学年龄的女学生的姓名和学号。

(11) 查询所有与刘宏伟同年级、同专业的学生的姓名、学号和性别。

(12) 查询选修 C001 号课程的学生中成绩最高的学生的学号。

(13) 查询学生姓名及其所选修课程的课程号和成绩。

(14) 查询选修两门以上课程的学生平均成绩(不及格的课程不参与统计)，并要求按平均成绩的降序排列出来。

(15) 求每个学生的平均成绩，只取前五名。

(16) 查询每个学生的总学分。

(17) 查询每门课成绩最低的学生的学号和课程号。

2. 利用 T-SQL 语句，对 bankcard 数据库完成下列查询。

(1) 查询账号为 4125428003351201000 的账户的交易信息。

(2) 查询银行卡有哪些种类，只显示种类列，过滤掉重复行。

(3) 查询身份证号为 133 *** 198708150101 的储户的基本信息及其所有账户的信息。

(4) 查询身份证号为 133 *** 198708150101 的储户的总存款余额。

(5) 求每个储户的账户个数和总存款余额，要求包括身份证号和姓名。

(6) 查询总存款余额最多的储户信息。

(7) 查询总存款余额超过 10000 的储户信息。

(8) 查询有效期在一个月以内的账户及相应的储户信息。

(9) 查询账号为 4367428003351209000 的账户 2016 年 8 月的总收入。

(10) 查询所有账号以 4367 开头的账户的信息，并按余额降序排列。

(11) 查询没有信用卡的储户的信息。

(12) 查询所有 VIP 储户的总存款余额。

(13) 将余额少于 500 的非 VIP 储户的账户的余额减 10。

第 5 章　T-SQL 编程

T-SQL 提供了一些特殊关键字,用于控制 T-SQL 语句、语句块和存储过程的执行流。在数据库开发过程中,函数起着很重要的作用,函数是由一个或多个 T-SQL 语句组成的子程序,可用于封装代码以便重新使用。

本章首先介绍 T-SQL 语言编程用到的基础知识,如标识符、变量、运算符、表达式、批处理、注释等内容,然后介绍 T-SQL 中的流程控制语句,最后介绍 T-SQL 编程中函数的应用。

5.1　T-SQL 编程基础

5.1.1　标识符

标识符是用来标识事物的符号,其作用类似于给事物取的名字。标识符分为两类:常规标识符和分隔标识符。

1. 常规标识符

常规标识符格式的规则如下:

(1) 常规标识符必须以汉字、英文字母(包括从 a～z 和从 A～Z 的字符以及其他语言的字母字符)、下画线_、@或#开头,后续字符可以是汉字、英文字母、基本拉丁字符或其他国家/地区字符中的十进制数字、下画线_、@、#。

(2) 常规标识符不能是 SQL Server 保留字,SQL Server 保留字不区分字母大小写。

(3) 常规标识符最长不能超过 128 个字符。

2. 分隔标识符

符合所有常规标识符格式规则的标识符可以使用分隔标识符,也可以不使用分隔标识符。不符合常规标识符格式规则的标识符必须使用分隔标识符。

分隔标识符括在中括号[ ]或双引号" "中。

在下列情况下,需要使用分隔标识符:

(1) 使用保留关键字作为对象名或对象名的一部分。

(2) 标识符的命名不符合常规标识符格式的规则。

5.1.2　变量

1. 变量的分类

变量可以分为两类:全局变量和局部变量。

1）全局变量

全局变量由系统提供且预先声明，通过在名称前加两个@符号区别于局部变量。用户只能使用全局变量，不能对它们进行修改。全局变量的作用范围是整个 SQL Server 系统，任何程序都可以随时调用它们。

2）局部变量

变量是程序设计语言中必不可少的组成部分，可以用它保存程序运行过程中的中间值，也可以用它在语句之间传递数据。T-SQL 语言中的变量是可以保存单个特定类型的数据值的对象，也称为局部变量，只在定义它们的批处理或过程中可见。

2. 局部变量定义

T-SQL 语言中的变量在定义和引用时要在其名称前加上标志@，而且必须先用 DECLARE 命令定义后才可以使用。其定义的一般格式如下：

```
DECLARE {@local_variable  data_type} [,…n]
```

各个参数的含义如下：

① @local_variable——用于指定变量的名称，变量名必须以符号@开头，并且变量名必须符合 SQL Server 的命名规则。

② data_type——用于设置变量的数据类型及其大小。data_type 可以是任何由系统提供的或用户定义的数据类型。但是，变量不能是 text、ntext 或 image 数据类型。

3. 局部变量的赋值方法

使用 DECLARE 命令声明并创建变量之后，系统会将其初始值设为 NULL，如果想要设定变量的值，必须使用 SET 命令或者 SELECT 命令。

```
SET { @local_variable = expression }
```

或者

```
SELECT { @local_variable = expression } [ ,…n ]
```

其中，参数@local_variable 是给其赋值并声明的变量，expression 是有效的 SQL Server 表达式。

4. 局部变量的作用域

一个变量的作用域就是可以引用该变量的 T-SQL 语句范围。

局部变量的作用域是从声明它们的地方开始到声明它们的批处理或存储过程的结尾。换言之，局部变量只能在声明它们的批处理或存储过程中使用，一旦这些批处理或存储过程结束，局部变量将自行清除。

5. 变量使用举例

【例 5-1】 创建了一个变量@CurrentDateTime，然后将 GETDATE()函数的值放在变量中，最后输出@CurrentDateTime 变量的值。

实现的步骤如下：

① 打开 SSMS，在窗口上部的工具栏的左侧找到“新建查询”按钮，单击该按钮。

② 输入要让 SQL Server 执行的 T-SQL 语句，这里输入下面列出的 T-SQL 语句。

```
DECLARE @CurrentDateTime char(30)
```

```
SELECT @CurrentDateTime = GETDATE()
SELECT @CurrentDateTime AS '当前的日期和时间'
GO
```


③ 单击工具栏中的“执行”按钮，当系统给出的提示信息为“命令已成功完成”时，说明执行成功。运行结果如图 5-1 所示。

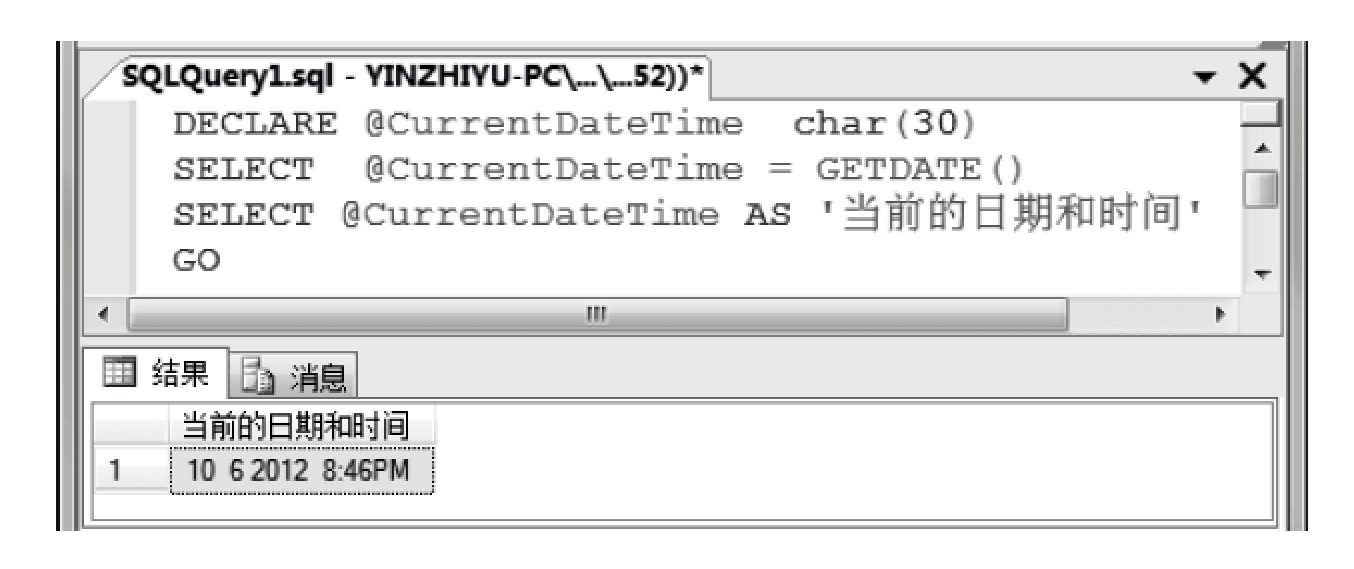


图 5-1　变量使用举例

注意：变量只在定义它的批处理中有效，因此，在上例中的程序中间不能写入 GO 语句。

5.1.3　运算符

运算符是一种符号，用来指定要在一个或多个表达式中执行的操作。在 Microsoft SQL Server 2008 系统中，可以使用的运算符可以分为算术运算符、逻辑运算符、赋值运算符、字符串连接运算符、位运算符、一元运算符及比较运算符等。

1. 算术运算符

(1) 算术运算符包括加(+)、减(－)、乘(*)、除(/)和取模(%)。

(2) 对于加、减、乘、除这四种算术运算符，计算的两个表达式可以是任何数据类型。

(3) 对于取模运算符，要求进行计算的数据的数据类型为 int、smallint 和 tinyint，完成的功能是返回一个除法运算的整数余数。

【例 5-2】　计算表达式的值，并将结果赋给变量@ExpResult。

程序清单如下：

```
DECLARE @ExpResult numeric
SET @ExpResult = 67 % 31
SELECT @ExpResult AS '表达式计算结果'
```

运行结果如图 5-2 所示。

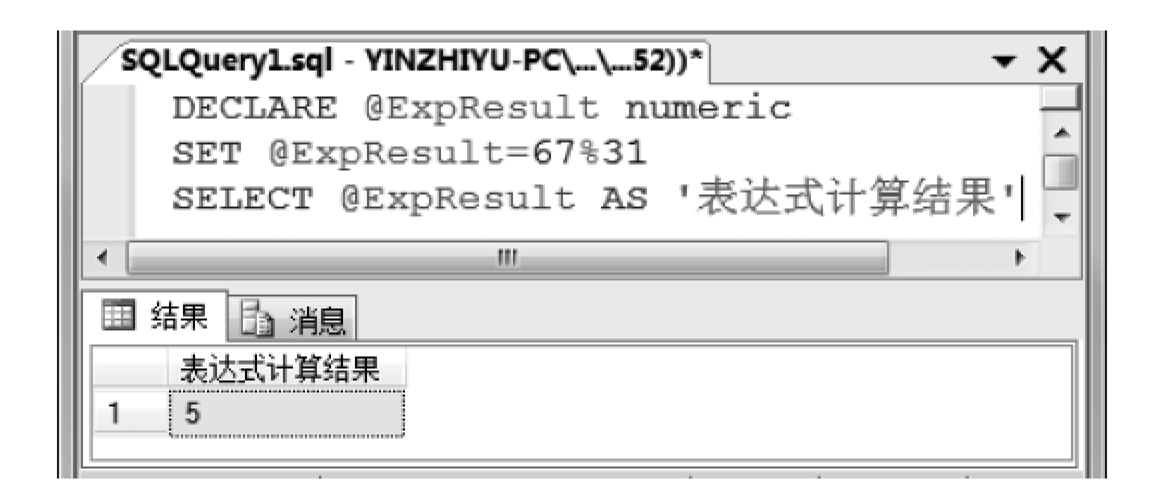


图 5-2　算术运算符例

2. 赋值运算符

T-SQL 中只有一个赋值运算符，即等号(＝)。赋值运算符使我们能够将数据值指派给特定的对象。另外，还可以使用赋值运算符在列标题和为列定义值的表达式之间建立关系。

【例 5-3】 创建一个@MyCounter 变量，然后赋值运算符将@MyCounter 设置为表达式返回的值。

```
DECLARE @MyCounter int
SET @MyCounter = 10
```

3. 位运算符

运算符包括按位与(&)、按位或(|)、按位异或(^)。

位运算符用来在整型数据或者二进制数据(image 数据类型除外)之间执行位操作。要求在位运算符左右两侧的操作数不能同时是二进制数据。位运算的运算规则如表 5-1 所示。

表 5-1 位运算的运算规则

运算符	运 算 规 则
&	两个位均为 1 时，结果为 1，否则为 0
|	只要一个位为 1，结果为 1，否则为 0
^	两个位值不同时，结果为 1，否则为 0

【例 5-4】 定义变量@a1 和@a2，给变量赋值，然后求两个变量与、或、异或的结果。

```
DECLARE @a1 int, @a2 int
SET @a1 = 3
SET @a2 = 8
SELECT @a1 & @a2 as 与, @a1 | @a2 as 或, @a1 ^@a2 as 异或
```

运行结果如图 5-3 所示。

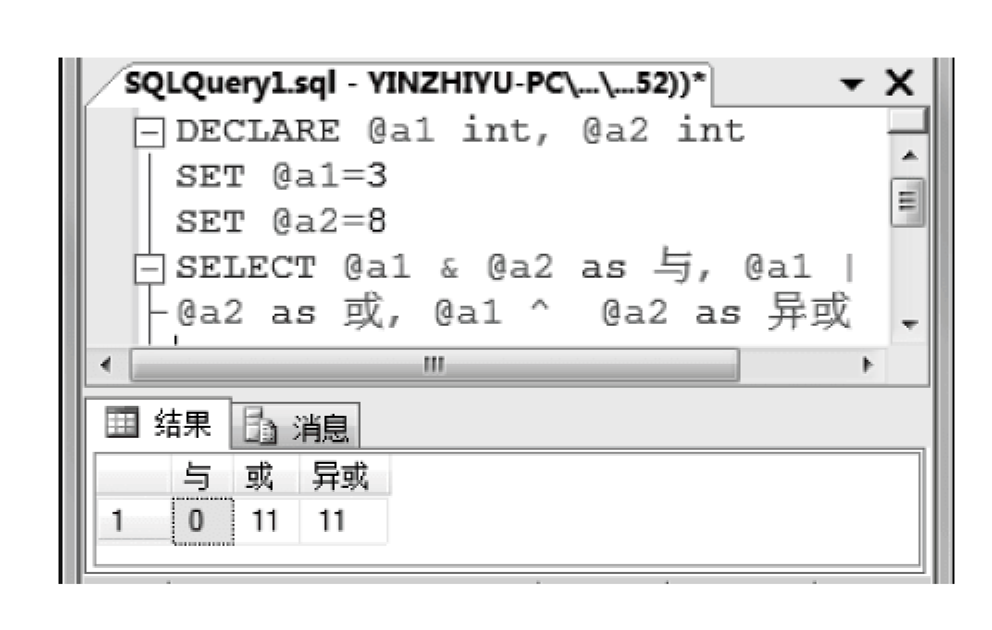


图 5-3 位运算符例

4. 比较运算符

比较运算符(又称关系运算符)见表 5-2，用于测试两个表达式的值是否相同，其运算结果为逻辑值，可以为三种之一：TRUE、FALSE 及 UNKNOWN(NULL 数据参与运算时)。

表 5-2　关系运算符及含义

运算符	含义	运算符	含义
=	相等	<=	小于或等于
>	大于	<>、!=	不等于
<	小于	!<	不小于
>=	大于或等于	!>	不大于

【例 5-5】　使用比较运算符计算表达式的值。

```
DECLARE @Exp1 int, @Exp2 int
SET @Exp1 = 30
SET @Exp2 = 50
IF @Exp1 >@Exp2
SELECT @Exp1 AS 小数据
```

运行结果如图 5-4 所示。

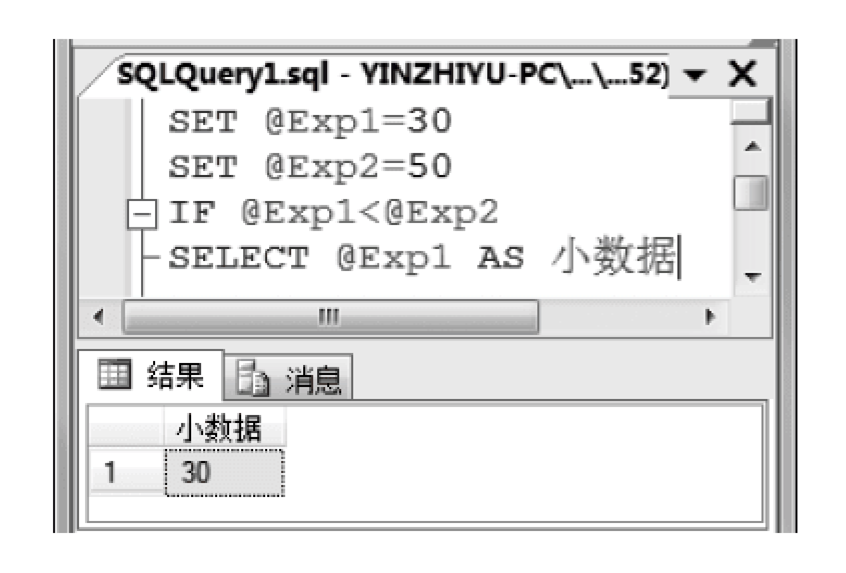


图 5-4　比较运算符例

5. 逻辑运算符

逻辑运算符对某些条件进行测试，以获得其真实情况。逻辑运算符和比较运算符一样，返回带有 TRUE 或 FALSE 值的 Boolean 数据类型或 UNKNOWN 值。逻辑运算符见表 5-3。

表 5-3　逻辑运算符

运　算　符	含　　义
ALL	如果一组的比较都为 TRUE，则为 TRUE
AND 或 &&	如果两个布尔表达式都为 TRUE，则为 TRUE
ANY 或 SOME	如果一组的比较中任何一个为 TRUE，则为 TRUE
BETWEEN	如果操作数在某个范围之内，则为 TRUE
EXISTS	如果子查询包含一些行，则为 TRUE
IN	如果操作数等于表达式列表中的一个，则为 TRUE
LIKE	如果操作数与一种模式相匹配，则为 TRUE
NOT 或 !	对任何其他布尔运算符的值取反
OR 或 ||	如果两个布尔表达式中的一个为 TRUE，则为 TRUE

6. 字符串连接运算符

连接运算符(+)用于两个字符串数据的连接，通常也称为字符串运算符。

在 SQL Server 中，对字符串的其他操作通过字符串函数进行。字符串连接运算符的操作数类型有 char、varchar、nchar、nvarchar、text、ntext 等。

【例 5-6】　使用字符串连接运算符计算表达式的值。

```
DECLARE @ExpResult char(60)
SET @ExpResult = '河北省石家庄市' + '河北师范大学' + '网络空间安全系'
SELECT @ExpResult AS '字符串的连接结果'
```

运行结果如图 5-5 所示。

7. 一元运算符

一元运算符只对一个表达式执行操作，该表达式可以是数值数据类型类别中的任何一

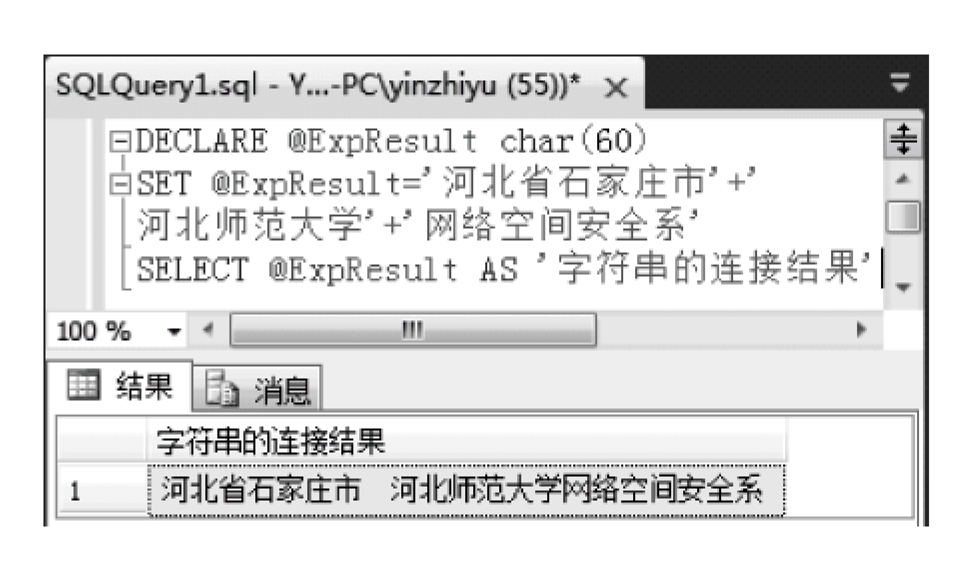


图 5-5 字符串连接运算符例

种数据类型。具体为：+(正)——数值为正；-(负)——数值为负；~(位非)——返回数字的非。其中，+(正)和-(负)运算符可以用数值数据类型类别中任一数据类型的任意表达式。~(位非)运算符只能用于整数数据类型类别中任一数据类型的表达式。

8. 运算符优先级和结合性

表达式计算器支持的运算符集中的每个运算符在优先级层次结构中都有指定的优先级，并包含一个计算方向，运算符的计算方向就是运算符结合性。具有高优先级的运算符先于低优先级的运算符进行计算。如果复杂的表达式有多个运算符，则运算符优先级将确定执行操作的顺序，执行顺序可能对结果值有明显的影响。

某些运算符具有相等的优先级。如果表达式包含多个具有相等优先级的运算符，则按照从左到右或从右到左的方向进行运算。表 5-4 按从高到低的顺序列出了运算符的优先级，同层运算符具有相等的优先级。

表 5-4 运算符的优先级与结合性

运算符	运算类型	结合性	运算符	运算类型	结合性
( )	表达式	从左到右	&	位与	从左到右
-，!，~	一元	从右到左	^	位异或	从左到右
cast as	一元	从右到左	|	位或	从左到右
*，/，%	乘法性的	从左到右	&&	逻辑与	从左到右
+，-	加法性的	从左到右	||	逻辑或	从左到右
<,>,<=,>=	关系	从左到右	? :	条件表达式	从右到左
==，!=	等式	从左到右			

5.1.4 批处理

批处理是包含一个或多个 T-SQL 语句的集合，从应用程序一次性地发送到 SQL Server 2008 进行执行，因此可以节省系统开销。SQL Server 将批处理的语句编译为一个可执行单元，称为执行计划，批处理的结束符为 GO。

编译错误(如语法错误)可使执行计划无法编译，因此未执行批处理中的任何语句。

运行时错误(如算术溢出或违反约束)会产生以下两种影响之一：

(1) 大多数运行时错误将停止执行批处理中当前语句和它之后的语句；

(2) 某些运行时错误(如违反约束)仅停止执行当前语句，而继续执行批处理中其他所有语句。

在遇到运行时错误之前执行的语句不受影响。唯一的例外是如果批处理在事务中而且

错误导致事务回滚，在这种情况下，回滚运行时错误之前所进行的未提交的数据修改。

5.1.5 注释

注释，也称为注解，是写在程序代码中的说明性文字，它们对程序的结构及功能进行文字说明。注释内容不被系统编译，也不被程序执行。

在 T-SQL 中可使用两类注释符：

(1) ANSI 标准的注释符“--”用于单行注释；

(2) 与 C 语言相同的程序注释符号，即“/ * …… * /”，“/ * ”用于程序注释开头，“ * /”用语程序注释结尾，可以在程序中将多行文字标示为注释。

批处理中的注释没有最大长度限制，一条注释可以包含一行或多行。下面是一些有效注释的示例。

```
USE teaching
-- 查询学生的学号和姓名
SELECT sno,sname FROM student
GO
/* 查询所有男同学的
学号、姓名和专业 */
SELECT sno,sname,specialty FROM student
WHERE ssex = '男'
GO
```

5.2 流程控制语句

与所有的计算机编程语言一样，T-SQL 也提供了用于编写过程性代码的语法结构，可用来进行顺序、分支、循环、存储过程等程序设计，编写结构化的模块代码，从而提高编程语言的处理能力。

SQL Server 提供的流程控制语句如表 5-5 所示。

表 5-5 流程控制语句

控制语句	说明	控制语句	说明
SET	赋值语句	CONTINUE	重新开始下一次循环
BEGIN…END	定义语句块	BREAK	退出循环
IF…ELSE	条件语句	GOTO	无条件转移语句
CASE	多分支语句	RETURN	无条件退出语句
WHILE	循环语句		

5.2.1 SET 语句

声明一个局部变量后，该变量将被初始化为 NULL。使用 SET 语句将一个不是 NULL 的值赋给声明的变量，给变量赋值的 SET 语句返回单值。在初始化多个变量时，为每个局部变量使用单独的 SET 语句。其语法格式为：

```
SET @local_variable = expression
```

说明：SET 语句是顺序执行的，将一个表达式赋值给声明的变量。表达式的数据类型必须和变量声明的类型相符。

【例 5-7】 声明变量，并用 SET 给变量赋值。

```
DECLARE @myvar char(20);
SET @myvar  =  'This is a test';
SELECT @myvar;
GO
```

除赋值外，SET 语句也实现一些设置功能，如：设置日期型数据的格式、设置数据库的某些属性等。

5.2.2 BEGIN...END 语句

BEGIN...END 语句能够将多个 T-SQL 语句组合成一个语句块，并将它们视为一个单元处理。其语法格式如下：

```
BEGIN
{ sql_statement  | statement_block  }
END
```

其中，参数{ sql_statement | statement_block }为任何有效的 T-SQL 语句或语句块。

5.2.3 IF...ELSE 语句

在程序中如果要对给定的条件进行判定，当条件为真或假时分别执行不同的 T-SQL 语句，可用 IF...ELSE 语句实现。

语法格式如下：

```
   IF Boolean_expression        /* 条件表达式，可含有 SELECT 语句 */
{ sql_statement | statement_block }
    /* 条件表达式为真时执行，语句块使用 BEGIN...END */
[ ELSE
{ sql_statement | statement_block } ]
   /* 条件表达式为假时执行，语句块使用 BEGIN...END */
```

其中，条件表达式的值必须是逻辑值，ELSE 子句是可选的。如果条件表达式中含有 SELECT 语句，则必须用小括号将 SELECT 语句括起来。

【例 5-8】 如果 C001 号课的平均成绩高于 80 分，则显示“C001 号课的平均成绩还不错”，否则显示“C001 号课的平均成绩一般”。

```
USE teaching
GO
IF ( SELECT AVG(score) FROM sc WHERE cno = 'C001' ) > 80
  PRINT 'C001 号课的平均成绩还不错'
ELSE
  PRINT 'C001 号课的平均成绩一般'
```

本例执行结果如图 5-6 所示。

【例 5-9】 输出 201502001 号学生的平均成绩，如果没有这个学生或该学生没有选课，则显示相应的提示信息。

```
USE teaching
GO
IF EXISTS ( SELECT * FROM SC WHERE sno = '201502001')
SELECT AVG(score) AS '201502001 号学生的平均分' FROM sc
WHERE sno = '201502001'
ELSE
  IF EXISTS ( SELECT * FROM student WHERE sno = '201502001')
PRINT '201502001 号学生没选课'
ELSE PRINT '没有 201502001 号学生'
```

本例执行结果如图 5-7 所示。

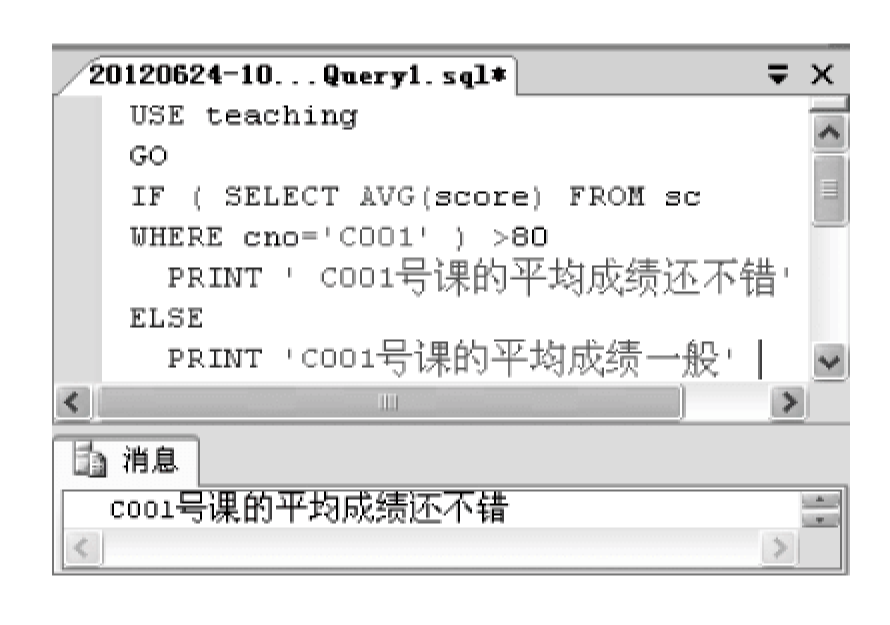


图 5-6 例 5-8 的执行结果



图 5-7 例 5-9 的执行结果

与普通高级语言一样，T-SQL 中的 IF...ELSE 语句也可以嵌套。虽然没有嵌套层数的限制，但最好不要超过 3 层，否则会影响程序的可读性，造成修改复杂等。

5.2.4 CASE 语句

使用 CASE 语句可以进行多个分支的选择。

CASE 具有以下两种格式。

简单 CASE 格式：将某个表达式与一组简单表达式进行比较，以确定结果。

搜索 CASE 格式：计算一组布尔表达式，以确定结果。

（1）简单 CASE 的语法格式：

```
CASE input_expression
WHEN when_expression THEN result_expression
[ ...n ]
[ ELSE else_result_expression]
END
```

（2）搜索 CASE 的语法格式：

```
CASE
  WHEN Boolean_expression THEN result_expression
   [ ...n ]
  [ ELSE else_result_expression]
```

```
END
```

其中的参数说明如下：

① input_expression——是使用简单 CASE 格式时所计算的表达式。Input_expression 是任何有效的 Microsoft SQL Server 表达式。

② WHEN when_expression——是使用简单 CASE 格式时 input_expression 所比较的简单表达式。When_expression 是任意有效的 SQL Server 表达式。Input_expression 和每个 when_expression 的数据类型必须相同，或者进行隐性转换。

③ THEN result_expression——当 input_expression = when_expression 取值为 TRUE，或者 Boolean_expression 取值为 TRUE 时返回的表达式。result expression 是任意有效的 SQL Server 表达式。

④ n 占位符——表明可以使用多个 WHEN when_expression THEN result_expression 子句或 WHEN Boolean_expression THEN result_expression 子句。

⑤ ELSE else_result_expression——当比较运算取值不为 TRUE 时返回的表达式。如果省略此参数并且比较运算取值不为 TRUE，则 CASE 将返回 NULL 值。else_result_expression 是任意有效的 SQL Server 表达式。else_result_expression 和所有 result_expression 的数据类型必须相同，或者必须进行隐性转换。

⑥ WHEN Boolean_expression——是使用 CASE 搜索格式时所计算的布尔表达式。Boolean_expression 是任意有效的布尔表达式。

简单 CASE 格式的运行过程：

① 计算 input_expression，然后按指定顺序对每个 WHEN 子句的 input_expression=when_expression 进行计算。

② 返回第一个取值为 TRUE 的 input_expression = when_expression 的 result_expression。

③ 如果没有取值为 TRUE 的 input_expression = when_expression，则当指定 ELSE 子句时 SQL Server 将返回 else_result_expression；若没有指定 ELSE 子句，则返回 NULL 值。

搜索 CASE 格式的运行过程：

① 按指定顺序为每个 WHEN 子句的 Boolean_expression 求值。

② 返回第一个取值为 TRUE 的 Boolean_expression 的 result_expression。

③ 如果没有取值为 TRUE 的 Boolean_expression，则当指定 ELSE 子句时 SQL Server 将返回 else_result_expression；若没有指定 ELSE 子句，则返回 NULL 值。

【例 5-10】 以简单 CASE 格式查询所有学生的专业情况，包括学号、姓名和专业的英文名。

```
USE teaching
SELECT sno,sname,
    CASE specialty
    WHEN '计算机' THEN 'Computer'
    WHEN '电子信息' THEN 'Electronic Information'
    WHEN '通信工程' THEN 'Communication Engineering'
    ELSE 'Network Engineering'
```

```
END AS specialty
FROM student
```

执行结果如图 5-8 所示。

【例 5-11】 以搜索 CASE 格式查询所有学生的考试等级,包括学号、课程号和成绩级别(a、b、c、d、e)。

```
USE teaching
SELECT sno, cno,
 CASE
   WHEN   score >= 90 then 'a'
   WHEN   score >= 80 then 'b'
   WHEN   score >= 70 then 'c'
   WHEN   score >= 60 then 'd'
   WHEN   score < 60   then 'e'
 END AS score_level
FROM sc
```

执行结果如图 5-9 所示。

	sno	sname	specialty
1	201501001	张小玲	Electronic Information
2	201501008	张玲	Electronic Information
3	201502001	张强	Computer
4	201502003	张强	Electronic Information
5	201502005	刘梅	Computer
6	201602001	王一	Computer
7	201603005	郑丽	Network Engineering
8	201701003	沈艳	Electronic Information
9	201703001	赵丽红	Network Engineering
10	201704001	李宏伟	Communication Engin...
11	201704006	刘景鹏	Communication Engin...

图 5-8 简单 CASE 格式查询

	sno	cno	score_level
1	201501001	X003	NULL
2	201502005	X003	NULL
3	201602001	C001	a
4	201602001	C004	b
5	201602001	X003	b
6	201603005	C001	c
7	201603005	C004	a
8	201701003	C001	c

图 5-9 搜索 CASE 格式查询

5.2.5 WHILE 语句

如果需要重复执行程序中的一部分语句,可使用 WHILE 循环语句实现。WHILE 语句通过布尔表达式来设置一个条件,当这个条件成立时,重复执行一个语句或语句块,重复执行的部分称为循环体。可以使用 BREAK 和 CONTINUE 关键字在循环内部控制 WHILE 循环中语句的执行。

语法格式为:

```
WHILE  Boolean_expression                    /* 条件表达式 */
    sql_statement1 | statement_block1
    [BREAK]
    [sql_statement2 | statement_block2]
    [CONTINUE]
    [sql_statement3 | statement_block3]      /* T-SQL 语句序列构成的循环体 */
```

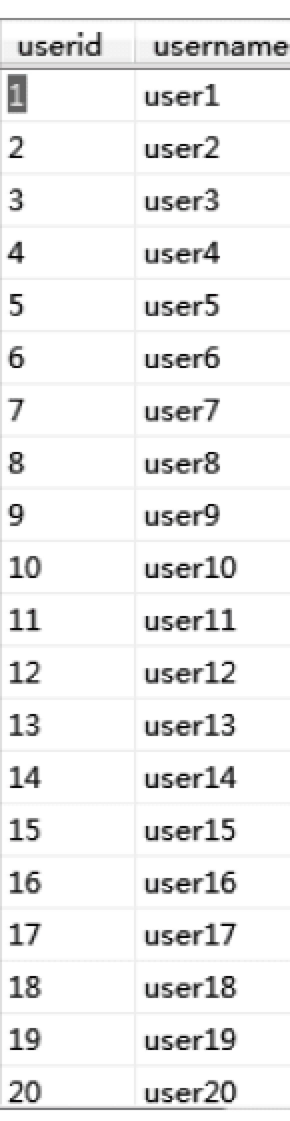
其中,BREAK 命令的功能是让程序跳出包含它的最内层循环,而 CONTINUE 命令可

以让程序跳过 CONTINUE 之后的语句回到 WHILE 循环的第一行命令。通常情况下，CONTINUE 和 BREAK 是放在 IF...ELSE 命令中的，即在满足某个条件的前提下提前结束本次循环或退出本层循环。WHILE 语句也可以嵌套。

【例 5-12】 创建一个 usern 表，包含 userid 和 username 列，接着利用 while 循环向其中插入前 20 行数据。

```
declare @i int
set @i = 1
while @i <= 20
begin
insert into usern (userid,username) values(@i, 'user' + ltrim(str
(@i)))
set @i = @i + 1
end
```

本例执行结果如图 5-10 所示。



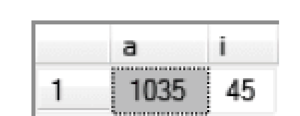
userid	username
1	user1
2	user2
3	user3
4	user4
5	user5
6	user6
7	user7
8	user8
9	user9
10	user10
11	user11
12	user12
13	user13
14	user14
15	user15
16	user16
17	user17
18	user18
19	user19
20	user20

图 5-10　WHILE 语句 1

【例 5-13】 求 1～100 的累加和，当和超过 1000 时停止累加，显示累加和以及累加到的位置。

```
DECLARE @i int,@a int
SET @i = 1
SET @a = 0
WHILE @i <=  100
  BEGIN
     SET @a = @a + @i
     IF @a >= 1000 BREAK
     SET @i = @i + 1
  END
SELECT @a AS 'a', @i AS 'i'
```

本例执行结果如图 5-11 所示。



	a	i
1	1035	45

图 5-11　WHILE 语句 2

5.2.6　GOTO 语句

GOTO 语句可以实现无条件的跳转。

语法格式为：

```
GOTO label                                /* label 为要跳转到的语句标号 */
```

其中，标号是 GOTO 的目标，它仅标识了跳转的目标。标号不隔离其前后的语句。执行标号前面语句的用户将跳过标号并执行标号后的语句。除非标号前面的语句本身是控制流语句（如 RETURN），这种情况才会发生。

【例 5-14】 用 GOTO 实现循环：求 1～100 的和。

```
DECLARE @s int,@i int
SET @i = 0
SET @s = 0
my_loop:
     SET @s = @s + @i
```

```
      SET @i = @i + 1
IF @i <= 100 GOTO my_loop
PRINT'1_2 + ... + 100 = ' + CAST(@s as char(25))
```


本例执行结果如图 5-12 所示。

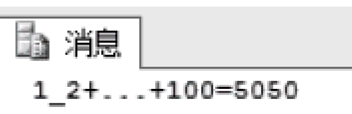


图 5-12　GOTO 语句 1

【例 5-15】　输出 201602001 号学生的平均成绩，若没有这个学生或该学生没选课，则显示相应的提示信息，用 GOTO 语句实现。

```
DECLARE @avg float
IF (SELECT  count( * ) FROM sc WHERE sno = '201602001') = 0
GOTO label1
BEGIN
  SELECT  @avg = avg(score)  FROM  sc WHERE sno = '201602001'
  PRINT '201602001 号学生的平均成绩: ' +  cast(@avg as varchar)
  RETURN
END
Label1: PRINT '没有 201602001 号学生或 201602001 号学生没选课'
```

一般来说，应尽量少使用 GOTO 语句。过多使用 GOTO 语句可能会使 T-SQL 批处理的逻辑难以理解。使用 GOTO 实现的逻辑几乎都可以使用其他控制流语句实现。GOTO 最好作为跳出深层嵌套的控制流语句。

5.2.7　RETURN 语句

使用 RETURN 语句，可以从查询或过程中无条件退出。可在任何时候用于从过程、批处理或语句块中退出，而不执行位于 RETURN 之后的语句。

语法格式为：

```
RETURN [integer_expression]                    /* 整型表达式 */
```

其中，整型表达式为一个整数值，是 RETURN 语句要返回的值。

注意：当用于存储过程时，不能返回空值。如果试图返回空值，将生成警告信息，并返回 0 值。

【例 5-16】　利用存储过程求某个学号学生的平均成绩。

```
USE teaching
GO
CREATE PROCEDURE mypro @no char(7)
AS   RETURN(SELECT  AVG(score) FROM sc WHERE sno =  @no)
```

创建查询：查询 201602001 号学生的姓名和平均成绩。

```
DECLARE @avg float,@no char(7)
SET @no = '201602001'
EXEC  @avg = mypro @no
SELECT sname, @avg as  '平均分'  FROM student WHERE sno = @no
```

本例执行结果如图 5-13 所示。

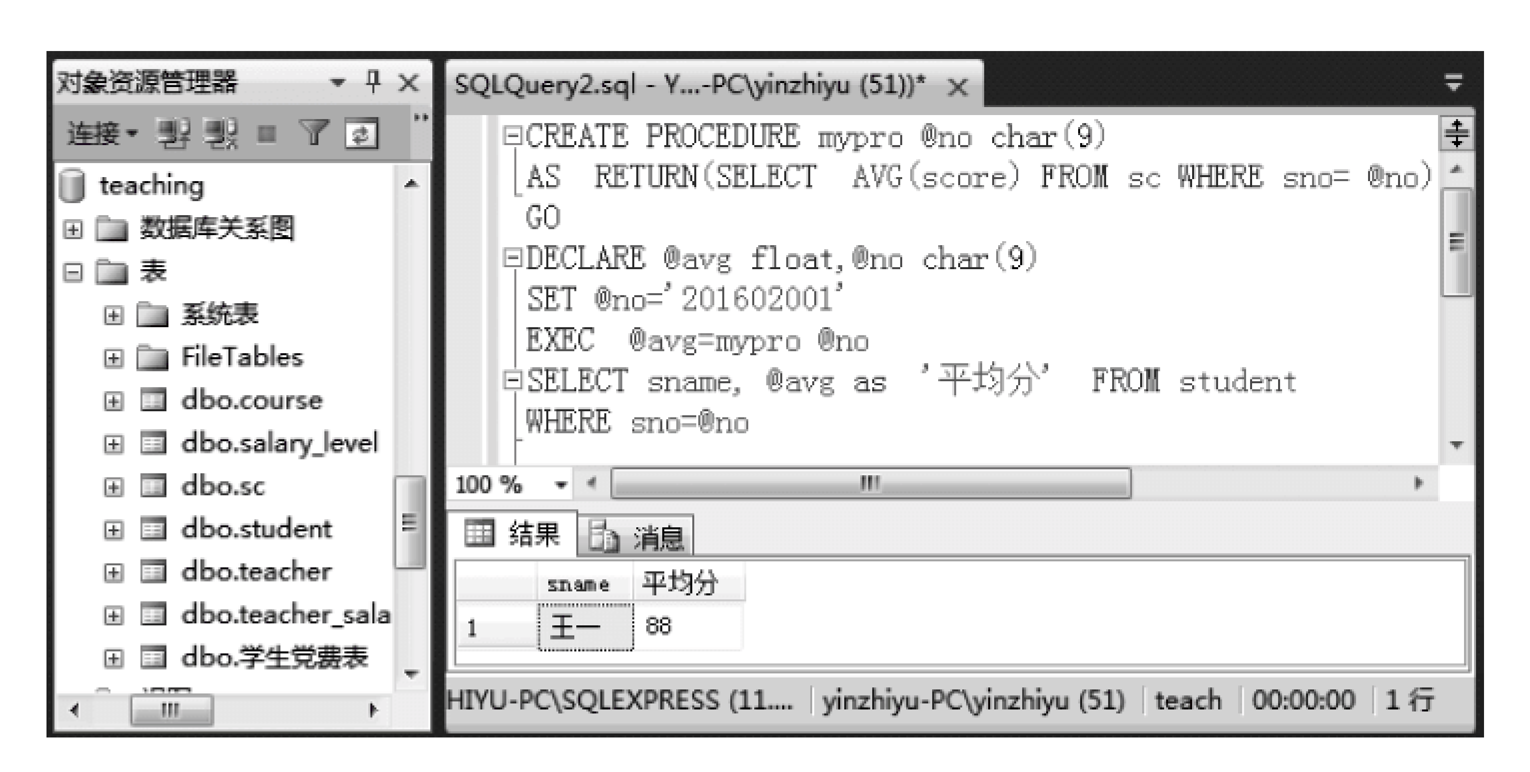


图 5-13　RETURN 语句例

5.3　函　　数

函数是由一个或多个 T-SQL 语句组成的子程序，可用于封装代码以便重新使用。T-SQL 语言提供了丰富的数据操作函数，用于完成各种数据管理工作。当然，SQL Server 并不将用户限制在定义为 T-SQL 语言一部分的内置函数上，而是允许用户创建自己的用户定义函数。

5.3.1　系统内置函数

在程序设计过程中，常常调用系统提供的函数，SQL Server 数据库管理人员必须掌握 SQL Server 的函数功能，并将 T-SQL 语言的程序或脚本与函数相结合，这将极大地提高数据管理工作的效率。

T-SQL 提供的内置函数按其值是否具有确定性可分为确定性的和非确定性的两大类。

(1) 确定性函数：每次使用特定的输入值集调用该函数时，总是返回相同的结果。

(2) 非确定性函数：每次使用特定的输入值集调用时，它们可能返回不同的结果。

例如，DATEADD 内置函数是确定性函数，因为对于其任何给定参数总是返回相同的结果。GETDATE 是非确定性函数，因其每次执行后，返回结果都不同。

T-SQL 系统内置函数按函数的功能分类可分为系统函数、聚合函数、数学函数、字符串函数、日期和时间函数、转换函数、排名函数、行集函数等类型。

1. 数学函数

SQL Server 2008 中提供了许多数学函数，可以满足数据库维护人员日常的数值计算需要，常用的数学函数见表 5-6。

表 5-6　SQL Server 2008 中常用的数学函数

函数名称	函数功能	函数名称	函数功能
ABS	求绝对值	POWER	求 x 的 y 次方
COS	余弦函数	RAND	求随机数
COT	余切函数	ROUND	四舍五入
EXP	计算 e 的 x 次幂	SIN	正弦函数
FLOOR	求仅次于最小值的值	SQUARE	开方
LOG	求自然对数	SQRT	求平方根
PI	常量，圆周率	TAN	正切函数

【例 5-17】 求下列语句的执行结果。

(1) SELECT FLOOR(10.9), FLOOR(−10.9)

(2) SELECT ROUND(10.9,0), ROUND(−10.9,0)

(1)执行结果为：10,−11

(2)执行结果为：11.0,−11.0

2. 日期和时间函数

SQL Server 2008 提供了众多的日期和时间函数,用于进行时间方面的处理工作。

在 datetime 类型的值上进行操作是常规的做法,例如,“获取当前日期”“计算 50 天后是什么日期”或者“指出特别的日期是星期几”。

以下列出了几个常用的日期和时间函数。

(1) GETDATE()：返回系统当前的日期和时间。

(2) DATEADD (datepart, integer_expression, date_expression)：返回指定日期 date_ expression(日期表达式)加上指定的额外日期间隔 integer_expression(整型表达式)产生的新日期。

(3) DATEDIFF (datepart, date_expression1, date_expression2)：返回两个指定日期在 datepart 方面的不同之处,即 date_expression2 超过 date_expression1 的差距值,其结果值是一个带有正负号的整数值。

(4) DATENAME (datepart,date_expression)：以字符串的形式返回日期的指定部分,此部分由 datepart 来指定。

(5) DATEPART (datepart, date_expression)：以整数值的形式返回日期表达式的指定部分。此部分由 datepart 来指定。

DATEPART()函数和 DATENAME()函数极其相似,只不过前者返回的是时间的名称,后者返回的是具体的时间数值。

(6) day(date_expression)：返回日期表达式中的日。

(7) month(date_expression)：返回日期表达式中的月。

(8) year(date_expression)：返回日期表达式中的年。

【例 5-18】 计算现在是几月。

```
SELECT MONTH (GETDATE())
```

3. 聚合函数

聚合函数在结果集中通过对被选列值的收集处理并返回一个数值型的计算结果,在 T-SQL 的数据查询中经常使用,此处不再赘述。

4. 字符串函数

SQL Server 2008 中的字符串函数也有很多,主要用来处理二进制类型的数据和文本类型的数据。以下列出了一些常用的字符串函数。

(1) ASCII (char_expression)：返回表达式中最左边一个字符的 ASCII 码值。

(2) CHAR(integer_expression)：返回整数所代表的 ASCII 码值对应的字符。

(3) LOWER(char_expression)：将大写字符转换为小写字符。

(4) UPPER(char_expression)：将小写字符转换为大写字符。

(5) LTRIM(char_expression)：删除字符串开始部分的空格。

(6) RTRIM(char_expression)：删除字符串尾部的空格。

(7) RIGHT(char_expression, integer_expression)：返回 char_expression 字符串中 integer_expression 个字符以后的部分字符串，integer_expression 为负时，返回 NULL。

(8) SPACE(integer_expression)：返回由 integer_expression 个空格组成的字符串，integer_expression 为负时，返回 NULL。

(9) STR(float_expression[,length[,decimal]])：将一个数值型数据转换为字符串，length 为字符串的长度，decimal 为小数点的位数。

(10) STUFF(char_expression1,start,length,char_expression2)：从 char_expression1 字符串的 start 个字符位置处删除 length 个字符，然后把 char_expression2 字符串插入到 char_expression1 的 start 处。

(11) SUBSTRING(expression,start,length)：从 expression 的第 start 个字符处返回 length 个字符。

(12) REVERSE(char_expression)：返回 char_expression 的逆序。

(13) CHARINDEX('pattern', char_expression)：返回指定 pattern 字符串在表达式中的起始位置。

【例 5-19】 将字符串"I am a student"以大写字母显示。

```
SELECT UPPER ('I am a student')
```

5.3.2 用户定义函数

用户定义函数可以针对特定应用程序问题提供解决方案，这些任务可以简单到计算一个值，也可能复杂到定义和实现数据表的约束。从技术上看，SQL Server 用户定义函数都是经过封装的 T-SQL 子程序，可以通过其他 T-SQL 代码调用这些子程序来返回单一的值或者数据表值。

在 SQL Server 中根据函数返回值形式的不同将用户自定义函数分为三种类型：标量函数、内嵌表值函数和多语句表值函数。

标量函数返回一个确定类型的标量值，其返回值类型为除 TEXT、NTEXT、IMAGE、CURSOR、TIMESTAMP 和 TABLE 类型以外的其他数据类型。函数体语句定义在 BEGIN...END 语句内，其中包含了可以返回值的 T-SQL 命令。

内嵌表值函数以表的形式返回一个返回值，即它返回的是一个表。内嵌表值函数没有由 BEGIN...END 语句括起来的函数体。其返回的表由一个位于 RETURN 子句中的 SELECT 命令段从数据库中筛选出来。内嵌表值函数的功能相当于一个参数化的视图。

多语句表值函数可以看作是标量型和内嵌表值函数的结合体。它的返回值是一个表，但它和标量函数一样有一个用 BEGIN...END 语句括起来的函数体，返回值的表中的数据是由函数体中的语句插入的。由此可见，它可以进行多次查询，对数据进行多次筛选与合并，弥补了内嵌表值型函数的不足。

1. 标量函数的创建与调用

创建标量函数的语法格式：

```
CREATE FUNCTION [ owner_name. ] function_name      /* 函数名部分 */
 ( [ { @parameter_name [AS] parameter_data_type
  [ = DEFAULT] } [ ,...n ] ] )                     /* 形参定义部分 */
RETURNS return_data_type                           /* 返回参数的类型 */
[ AS ]
BEGIN
   function_body                                   /* 函数体部分 */
   RETURN expression                               /* 返回语句 */
END
```

说明：owner_name 指定用户自定义函数的所有者；function_name 指定用户自定义函数的名称；@parameter_name 定义一个或多个参数的名称，一个函数最多可以定义 1024 个参数，参数的作用范围是整个函数；parameter_data_type 和 return_data_type 指定参数的数据类型和返回值的数据类型，二者都可以为除 TEXT、NTEXT、IMAGE、CURSOR、TIMESTAMP 和 TABLE 类型以外的其他数据类型；function_body：指定一系列的 T-SQL 语句，它们决定了函数的返回值；expression：指定用户自定义函数返回的标量值表达式。当函数的参数有默认值时，调用该函数时必须指定默认 DEFAULT 关键字才能获取默认值。

【例 5-20】 求选课表中某门课的平均成绩。

```
USE teaching
GO
CREATE FUNCTION average(@cn char(4)) RETURNS float
AS
BEGIN
   DECLARE @aver float
   SELECT @aver = ( SELECT avg(score) FROM sc WHERE cno = @cn)
   RETURN @aver
END
```

在其他程序模块中调用标量函数的语法格式：

```
owner_name.function_name(parameter_expression 1...parameter_expression n)
```

其含义为：所有者名.函数名(实参 1,…,实参 n)。当调用用户定义的标量函数时，必须提供至少由两部分组成的名称(所有者名.函数名)。可以在 SELECT 语句中调用，实参可为已赋值的局部变量或表达式。

【例 5-21】 求 C001 号课的平均成绩。

```
USE teaching
DECLARE @course1 char(4)
SET @course1 = 'C001'
SELECT  dbo.average(@course1)  AS  'C001 号课程的平均成绩'
```

本例执行结果如图 5-14 所示。

【例 5-22】 新建 course1 表，包括课程号(cno)、课程名(cname)、学分(credit)平均成绩(aver)，其中平均成绩列为计算列，调用例 5-20 定义的函数 average。

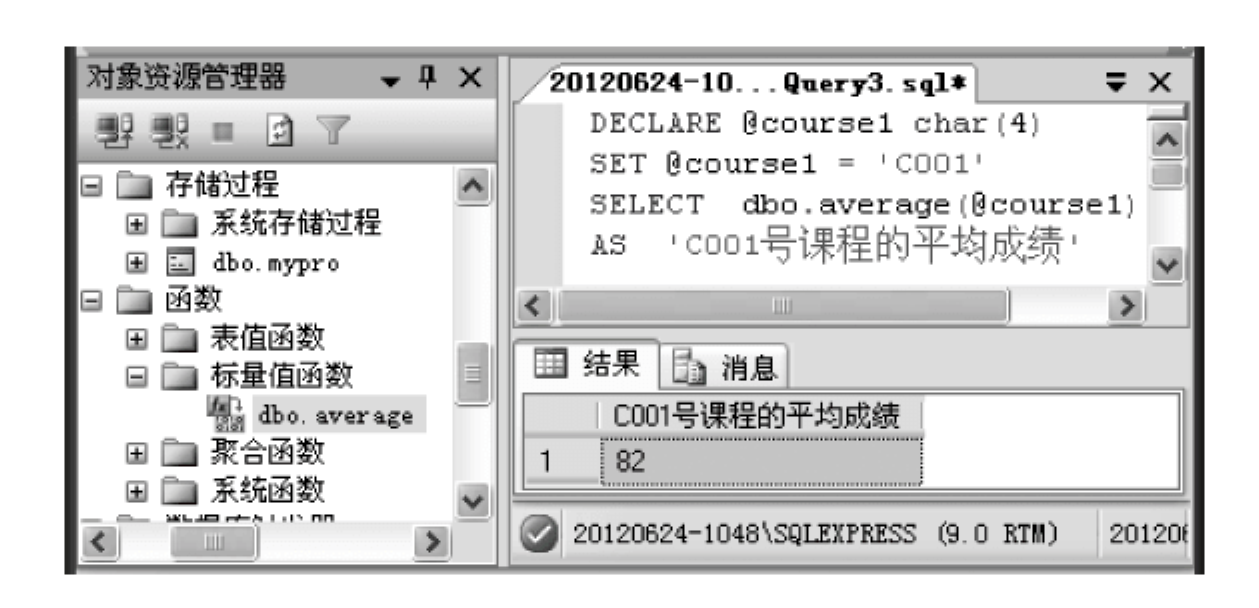


图 5-14　标量函数例

```
create table course1
(cno char(4) primary key,
cname nvarchar(20) ,
credit int ,
aver as (dbo.average(cno))
)
```

2. 内嵌表值函数的创建与调用

创建内嵌表值函数的语法格式：

```
CREATE FUNCTION [ owner_name. ] function_name       /*定义函数名部分*/
( [ { @parameter_name [AS] parameter_data_type
 [ = DEFAULT] }[ ,...n ] ] )                         /*定义参数部分*/
RETURNS table                                        /*返回值为表类型*/
[ AS ]  RETURN [ ( ]SELECT statement [ ) ]           /*通过 SELECT 语句返回内嵌表*/
```

说明：table 指定返回值为一个表；SELECT statement 指单个 SELECT 语句，确定返回的表的数据。其余参数与标量函数相同。

【例 5-23】 查询某个专业所有学生的学号、姓名、所选课程的课程号和成绩。

```
USE teaching
GO
CREATE FUNCTION st_func(@major nvarchar(10))  RETURNS  table
AS RETURN
    ( SELECT student.sno, student.sname,cno,score FROM student,sc
      WHERE specialty = @major AND student.sno = sc.sno)
```

因为内嵌表值函数的返回值为 table 类型，所以在其他程序模块中调用此类函数时，只能通过 SELECT 语句。

【例 5-24】 查询计算机专业所有学生的学号、姓名、所选的课程号和成绩。

```
USE teaching
GO
SELECT  *  FROM  st_func ('计算机')
```

本例执行结果如图 5-15 所示。

3. 多语句表值函数的创建与调用

内嵌表值函数和多语句表值函数都返回表，二者的不同之处在于：内嵌表值函数没有函数主体，返回的表是单个 SELECT 语句的结果集；而多语句表值函数在 BEGIN...END

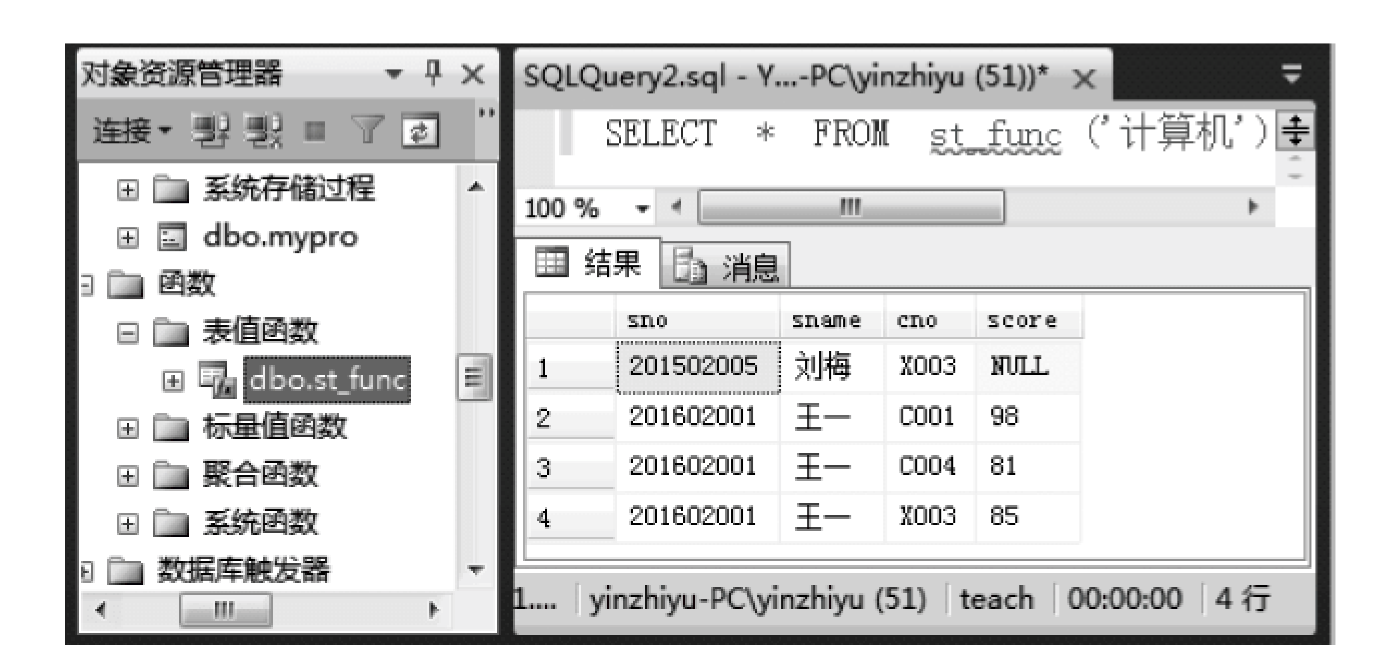


图 5-15　内嵌表值函数例

块中定义的函数主体包含 T-SQL 语句，这些语句可生成行，并将行插入至表中，最后返回表。

创建多语句表值函数的语法格式：

```
CREATE FUNCTION [ owner_name. ] function_name              /*定义函数名部分*/
  ( [ { @parameter_name [AS] parameter_data_type [ = DEFAULT] }[ ,...n ] ] )
/*定义函数参数部分*/
RETURNS @return_variable table < table_definition >      /*定义作为返回值的表*/
[ AS ]
BEGIN
  function_body                                           /*定义函数体*/
   RETURN
END
```

说明：@return_variable 是一个 TABLE 类型的变量，用于存储和累积返回的表中的数据行。其余参数与标量函数相同。

【例 5-25】 创建多语句表值函数，通过 sno 作为实参调用该函数，可显示该学生的姓名以及各门功课的成绩和学分。

```
CREATE FUNCTION st_score (@no char(9)) RETURNS @score table
( s_no char(9) ,
  s_name nvarchar(8) ,
  c_name nvarchar(20) ,
  c_score tinyint ,
  c_credit tinyint   )
AS BEGIN
     INSERT into @score
     SELECT s.sno,s.sname,c.cname, sc.score , c.credit
     FROM student s,course c,sc sc WHERE s.sno = sc.sno
     AND c.cno = sc.cno AND s.sno = @no
     RETURN
  END
```

多语句表值函数的调用与内嵌表值函数的调用方法相同，只能通过 SELECT 语句调用。

【例 5-26】 查询 201602001 号学生的姓名以及各门功课的成绩和学分。

```
SELECT * FROM st_score ('201602001')
```

本例执行结果如图 5-16 所示。

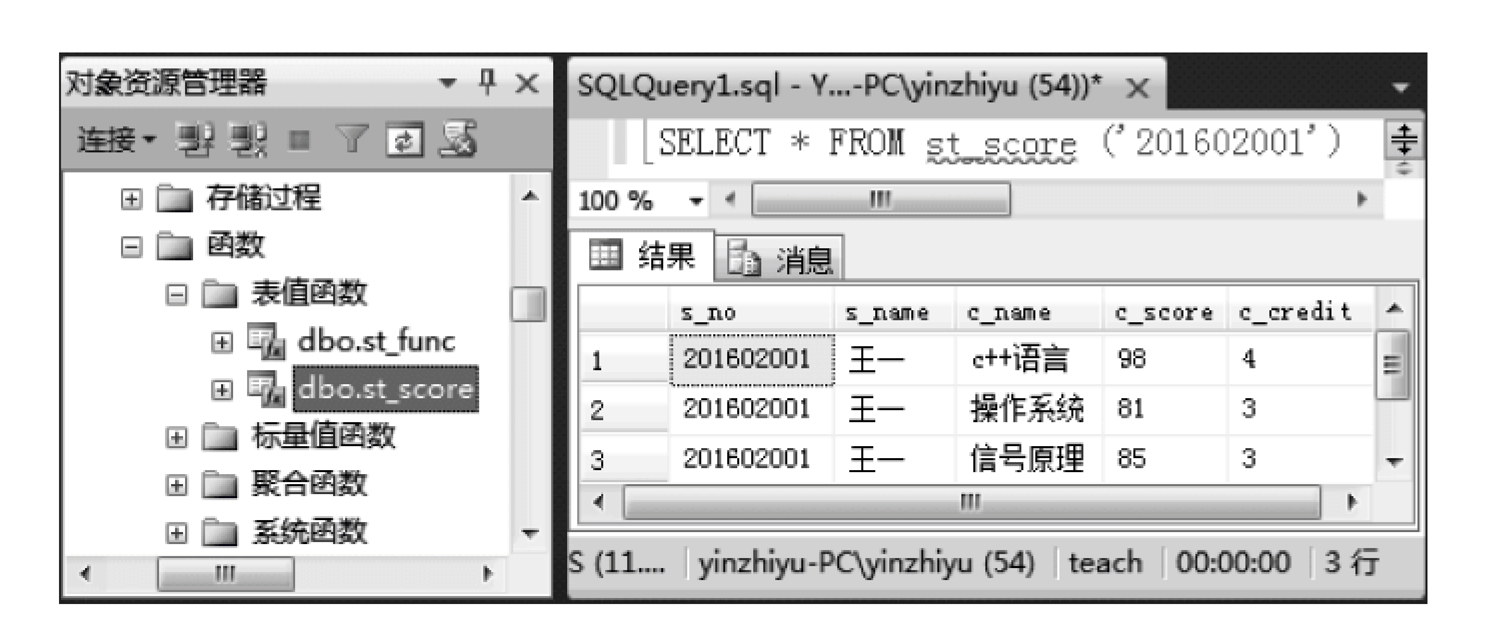


图 5-16 多语句表值函数例

习 题 5

1. 什么是标识符?
2. T-SQL 中有几种标识符? 它们的区别是什么?
3. 什么是局部变量? 什么是全局变量? 如何表示它们?
4. 以下变量名中,哪些是合法的变量名? 哪些是不合法的变量名?

A1,1a,@x,@@y,& 变量 1,@姓名,姓名,#m,##n,@@@abc##,@my_name

5. SQL Server 2008 所使用的运算符类别有哪些?
6. 利用 T-SQL 语句计算下列表达式,并给出运算结果。

(1) 9-3*5/2+6%4 (2)5&2|4 (3)'你们'+'好' (4)~10

7. 给出以下 T-SQL 语句的运行结果。

```
DECLARE @d SMALLDATETIME
SET @d = '2007 - 1 - 26'
SELECT @d + 10,@d - 10
```

8. 什么是批处理? 使用批处理有何限制? 批处理的结束符是什么?
9. 注释有几类? 它们分别是什么?
10. 针对 teaching 数据库,利用流程控制语句,查询学号为 201602001 的学生的各科成绩,如果没有这个学生的选课信息,则显示“此学生未选课”。
11. 针对 teaching 数据库,用函数实现:求某个专业选修了某门课的学生人数。然后,调用此函数。
12. 针对 teaching 数据库,用函数实现:查询某个专业学生所选的每门课的平均成绩。然后,调用此函数。
13. 针对 bankcard(银行卡管理)数据库中的 account 表,查询账户的等级,包括账号、身份证号、卡类型和等级(余额在 500 元以内为“小额账户”,500~10 000 元为“普通账户”,10 000 元以上为“大额账户”)。

第6章 视图和索引

数据库的基本表是由数据库设计人员根据所有用户的需求，按照规范化设计方法设计的，并不一定符合用户的应用需求。SQL Server 可以根据各个用户的应用需求重新定义表的数据结构，这种数据结构就是视图。索引是以表列为基础的数据库对象，它保存着表中排序的索引列，并且记录了索引列在数据表中的物理存储位置，实现了表中数据的逻辑排序。索引可以使数据库程序在最短的时间内找到所需要的数据，而不必查找整个数据库，这样可以节省时间，提高查找效率。

本章主要介绍视图的基本概念，视图的创建、修改和删除，利用视图实现对基本表中数据的各种操作，索引的基本概念，索引的分类，创建、修改和删除索引等操作。

6.1 视　　图

视图（View）是关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制。在用户看来，视图是通过不同路径去看一个实际表，就像一个窗口，我们通过窗口去看外面的高楼，可以看到高楼的不同部分，而透过视图可以看到数据库中自己感兴趣的内容。

6.1.1 视图概述

视图作为一种数据库对象，为用户提供了一个可以检索数据表中数据的方式。视图是一个虚表，可以视为另一种形式的表，是从一个或多个表中使用 SELECT 语句导出的虚表，那些用来导出视图的表称为基本表。

用户通过视图来浏览数据表中感兴趣的部分或全部数据，而数据的物理存储位置仍然在基本表中。所以视图并不是以一组数据的形式存储在数据库中，数据库中只存储视图的定义，而不存储视图对应的数据，这些数据仍存储在导出视图的基本表中，视图实际上是一个查询结果。当基本表中的数据发生变化时，视图中的数据也随之改变。

使用视图可以集中、简化和定制用户的数据库显示，用户可以通过视图来访问数据，而不必直接去访问该视图的基本表。

1. 视图的优点

(1) 为用户集中数据，简化用户的数据查询和处理。分散在多个表中的数据通过视图定义在一起，屏蔽了数据库的复杂性，用户不必输入复杂的查询语句，只需针对此视图做简单的查询即可。

(2) 保证数据的逻辑独立性。对于视图的操作,例如查询,只依赖于视图的定义,当构成视图的基本表需要修改时,只需要修改视图定义中的子查询部分,而基于视图的查询不用改变。

(3) 重新定制数据,使得数据便于共享。

(4) 提高了数据的安全性。对不同的用户可以定义不同的视图,使用户只能看到与自己有关的数据,简化了用户权限的管理,增加了数据的安全性。

2. 视图的分类

SQL Server 中,视图可以分为标准视图、索引视图和分区视图。

1) 标准视图

标准视图组合了一个或多个表中的数据,可以获得使用视图的大多数好处,可以实现对数据库的查询、修改和删除等基本操作。

2) 索引视图

索引视图是被具体化了的视图,它已经过计算并存储。可以为视图创建索引,即对视图创建一个唯一的聚集索引,这能显著提高某些类型查询的性能。索引视图尤其适于聚合许多行的查询,但不太适合于经常更新的基本数据集。

3) 分区视图

分区视图在一台或多台服务器间水平连接一组成员中的分区数据。这样,数据看上去如同来自于一个表。

6.1.2 创建视图

要使用视图,首先必须根据不同用户的应用需求创建视图。视图在数据库中是作为一个独立的对象存储的,必须遵循以下原则。

(1) 只能在当前数据库中创建视图。不过,如果使用分布式查询定义视图,则新视图所引用的表和视图可以存在于其他数据库中,甚至其他服务器上。

(2) 视图名称必须遵循标识符的规则,且对每个用户必须唯一。此外,该名称不得与该用户拥有的任何表的名称相同。

(3) 用户可以在其他视图之上建立视图。

(4) 如果视图中的某一列是由算术表达式、内置函数或常量派生而来,而且视图中两个或者更多的不同列拥有一个相同的名字(这种情况通常是因为在视图的定义中有一个连接,而且这两个或者多个来自不同表的列拥有相同的名字),此时,用户需要为视图的这些列指定特定的名称。

(5) 定义视图的查询不可以包含 ORDER BY 或 INTO 关键字。

(6) 不能在视图上进行全文索引定义。

(7) 不能创建临时视图,也不能在临时表上创建视图。

(8) 不能对视图执行全文查询,但是如果查询所引用的表支持全文索引,就可以在视图定义中包含全文查询。

(9) 不能将规则或者 DEFAULT 定义关联于视图。

在 SQL Server 服务器中创建视图主要有两种方式:一种方式是在 SSMS 中使用向导

创建视图；另一种方式是通过在查询窗口中执行 T-SQL 语句创建视图。

1. 在 SSMS 中创建视图

在 SSMS 中使用向导创建视图，是一种图形界面环境下最快捷的创建方式，其步骤如下：

(1) 在“对象资源管理器”中展开要创建视图的数据库，如 teaching，展开“视图”选项，可以看到视图列表中系统自动为数据库创建的系统视图。右击“视图”选项，选择“新建视图”命令，打开“添加表”对话框，在此对话框中，可以选择表、视图或者函数，然后单击“添加”按钮，就可以将其添加到视图查询中，如图 6-1 所示。

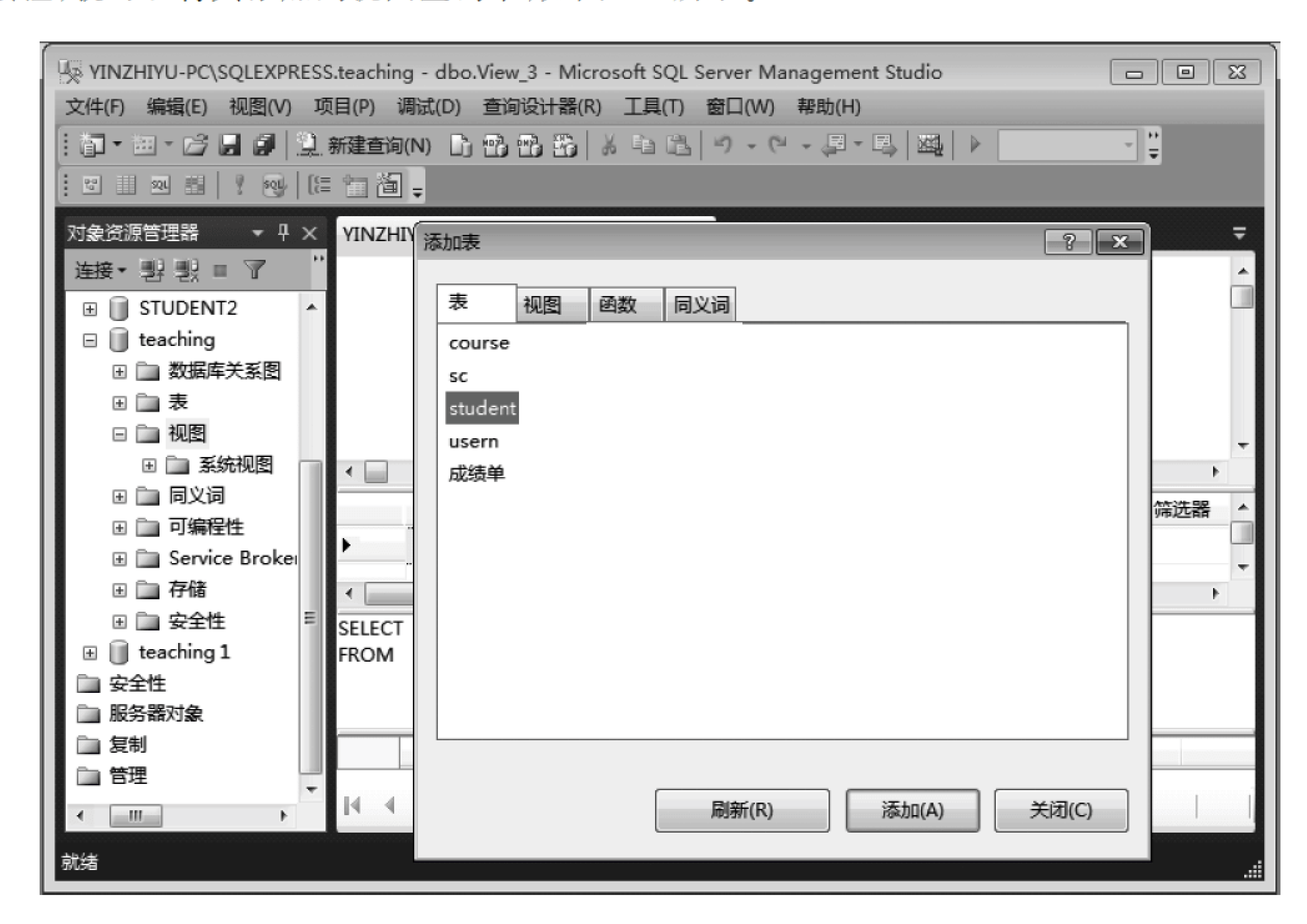


图 6-1　新建视图界面

(2) 以创建学生表中所有男生信息的视图为例。选择 student 表后，单击“添加”按钮，单击“关闭”按钮，返回“新建视图”对话框。

(3) 在对话框上半部分，可看到添加进来的 student 表，选择视图所用的列；在对话框中间网格窗格部分，可看到在上半部分的复选框中所选择的对应表的列，在 ssex 列的筛选器中写出筛选条件“ ='男'”；在对话框下半部分，可看到系统同时生成的 T-SQL 语句。然后，单击工具栏上的“保存”按钮，视图取名为 male_View，如图 6-2 所示。

在“对象资源管理器”中展开创建了视图的数据库，如 teaching，展开“视图”选项，就可以看到视图列表中刚创建好的男生视图 male_View，如图 6-3 所示。如果没有看到，则右击“视图”选项，单击快捷菜单中的“刷新”按钮。

2. 使用 T-SQL 语句创建视图

SQL Server 提供了 CREATE VIEW 语句创建视图，语法格式如下：

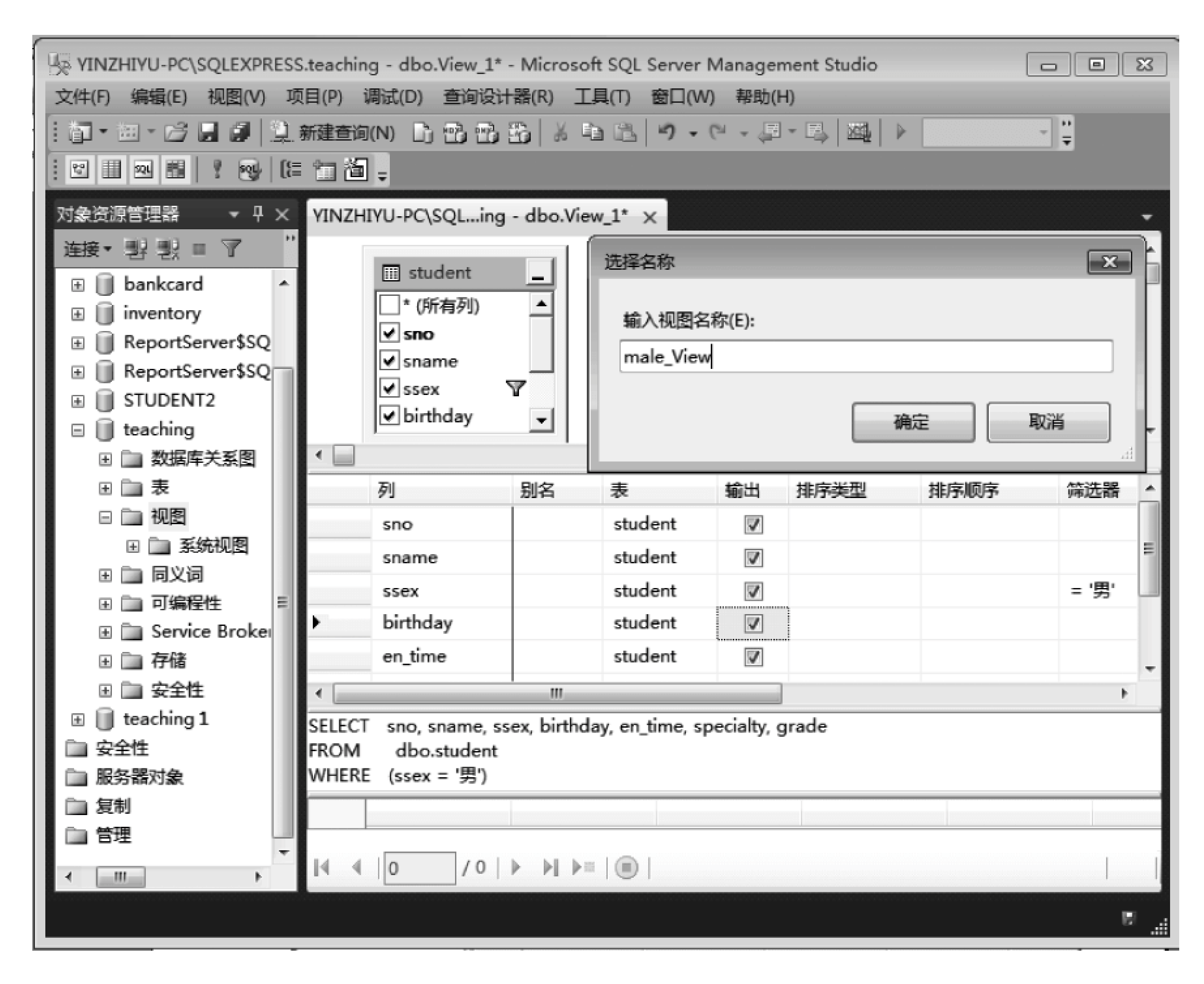


图 6-2　创建包含所有男生信息的视图



图 6-3　视图创建成功

```
CREATE VIEW [schema_name.]view_name [ (column_name[ ,...n ] ) ]
[with <view_attribute>[,...n]]
AS {select_statement}
[WITH CHECK OPTION]
```

其中各参数说明如下：

(1) schema_name——指定视图的所有者名称，包括数据库名和所有者名。

(2) view_name——视图的名称，视图名称必须符合标识符规则。

(3) column_name——视图中的列名。只有在下列情况下，才必须命名 CREATE VIEW 中的列：当列是从算术表达式、函数或常量派生的；两个或更多的列可能会具有相同的名称(通常是因为联接)；视图中的某列被赋予了不同于派生来源列的名称。如果未指

定 column_name，则视图列将获得与 SELECT 语句中的列相同的名称。

(4) with < view_attribute >——用于指定视图的属性。

视图的属性包括：

- ENCRYPTION——表示 SQL Server 加密包含 CREATE VIEW 语句文本的系统表列，可防止将视图作为 SQL Server 复制的一部分发布。
- SCHEMABINDING——将视图绑定到架构上。指定 SCHEMABINDING 时，select_ statement 必须包含所引用的表、视图或用户定义函数的两部分名称(owner. object，即拥有者. 对象名)。
- VIEW_METADATA——指定返回的结果是否为元数据。

(5) select_statement——就是定义视图的 SELECT 语句。该语句可以使用多个表或其他视图。视图不必是具体某个表的行和列的简单子集，可以用具有任意复杂性的 SELECT 子句，使用多个表或其他视图来创建视图。若要从创建视图的 SELECT 子句所引用的对象中选择，必须具有适当的权限。

(6) WITH CHECK OPTION——强制视图上执行的所有数据修改语句都必须符合由 select_statement 设置的准则。通过视图修改行时，WITH CHECK OPTION 可确保提交修改后，仍可通过视图看到修改的数据。

【例 6-1】 创建名为 s_c_sc 的视图，包括"计算机"专业的学生的学号、姓名及他们选修的课程号、课程名和成绩。

```
USE teaching
GO
CREATE VIEW s_c_sc
AS
SELECT student. sno, sname, course. cno, cname, score
FROM student, sc, course WHERE student. sno = sc. sno
AND course. cno = sc. cno AND specialty = '计算机'
GO
```

单击工具栏上的"!"按钮，执行 T-SQL 语句，视图创建成功，如图 6-4 所示。

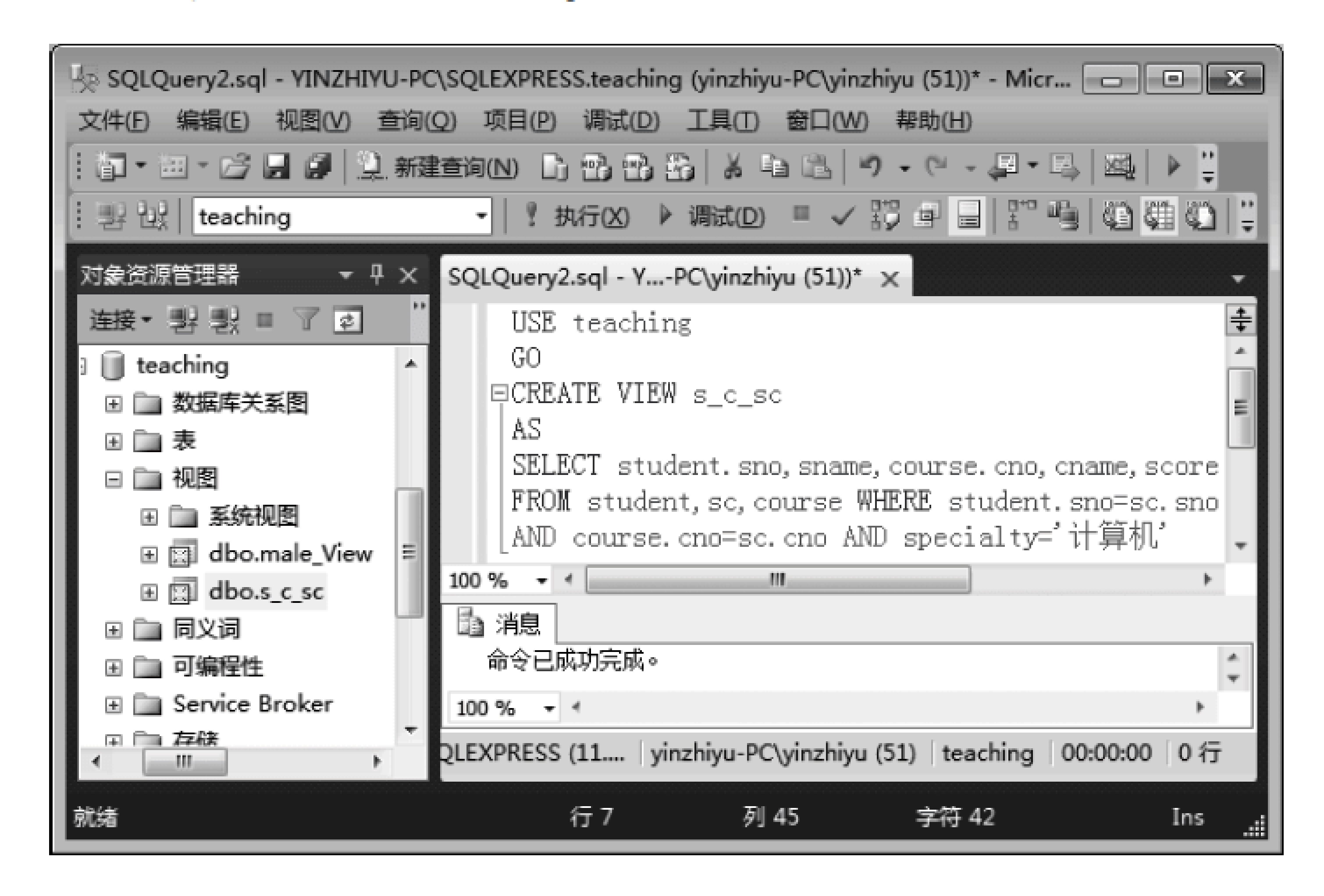


图 6-4 创建 s_c_sc 视图

【例 6-2】 创建名为 acc_count 的账户统计视图，求每个储户的账户个数，要求包括身份证号和姓名。

```
USE bankcard
GO
CREATE VIEW acc_count
AS
SELECT depositor.IDNO,Dname,COUNT( * ) AS Number
FROM depositor,account
WHERE depositor.IDNO = account.IDNO
GROUP BY depositor.IDNO,Dname
GO
```

单击工具栏上的“!”按钮，执行 T-SQL 语句，视图创建成功。

同在 SSMS 中创建的视图一样，在“对象资源管理器”中展开创建了视图的数据库，再展开“视图”选项，就可以看到视图列表中刚创建好的这两个视图。

6.1.3 修改视图

1. 在 SSMS 中修改视图

使用 SSMS 修改视图的操作步骤如下：

(1) 打开 SSMS 的“对象资源管理器”，展开相应数据库文件夹。

(2) 展开“视图”选项，右击要修改的视图，如 acc_count，选择“设计”命令，如图 6-5 所示，打开的对话框可用来修改视图的定义。

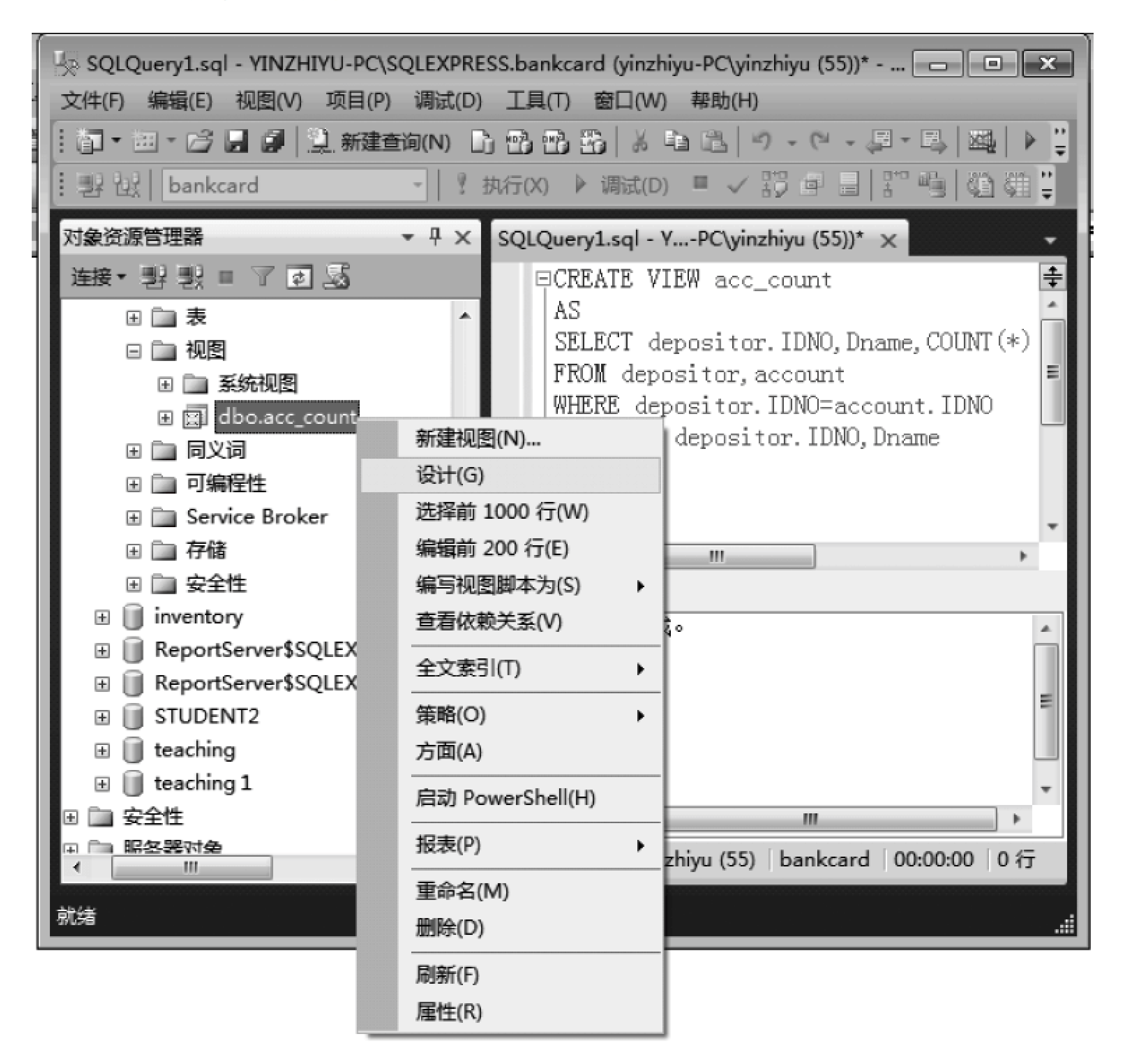


图 6-5 修改视图的定义

(3) 如果要向视图中再添加表，则可以在窗格中右击，选择“添加表”命令，如图 6-6 所示。如果要移除表，则右击要被移除的表，选择“移除”命令，如图 6-7 所示。




图 6-6　添加表菜单

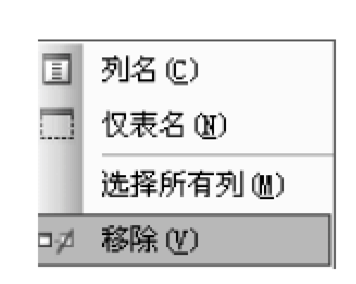


图 6-7　移除表菜单

(4) 如果要修改其他属性，则在对话框上半部分，可重新选择视图所用的列；在中间的网格部分，可对视图每一列进行属性设置。最后，单击工具栏上的“保存”按钮保存修改后的视图。

例如，将 male_View 视图修改为用于查询所有男生选课情况的信息。首先添加 sc 表，然后选择其中的 cno 和 score 列，其他属性不变。在对话框下半部分，可看到系统同时对 T-SQL 语句的修改，如图 6-8 所示。最后，单击工具栏上的保存的按钮保存修改后的 male_View 视图。

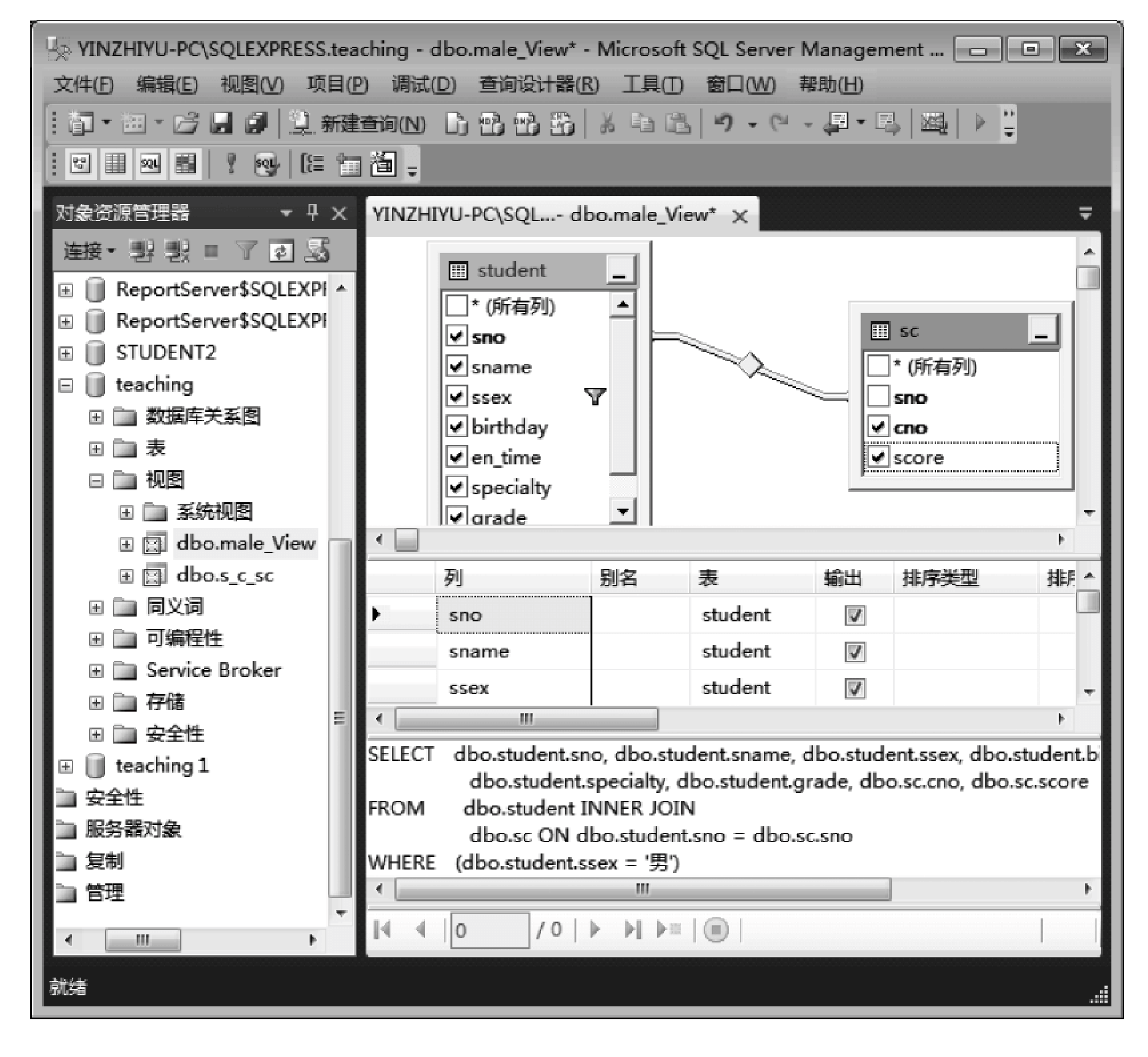


图 6-8　修改 male_View 视图

2. 使用 T-SQL 语句修改视图

T-SQL 提供了 ALTER VIEW 语句修改视图，语法格式如下：

```
ALTER VIEW [schema_name.]view_name
[ (column_name[ ,...n ] ) ] [with <view_attribute>[...n]]
AS {select_statement}
[ WITH CHECK OPTION ]
```

注：语句中的参数与 CREATE VIEW 语句中的参数相同。

【例 6-3】 修改 acc_count 视图，求每个储户的账户个数和总存款余额，要求包括身份证号和姓名。

```
USE bankcard
GO
ALTER VIEW acc_count
AS
SELECT depositor.IDNO,Dname,COUNT( * ) AS Number,SUM(Balance) SumBalance
FROM depositor,account
WHERE depositor.IDNO = account.IDNO
GROUP BY depositor.IDNO,Dname
GO
```

单击工具栏上的“!”按钮，执行 T-SQL 语句，视图修改成功。

【例 6-4】 在已有视图上创建新视图：创建 Few_Balance 余额统计视图，求总存款余额少于 5000 的储户信息，包括身份证号、姓名和手机号。

```
USE bankcard
GO
CREATE VIEW Few_Balance
AS
SELECT depositor.IDNO,depositor.Dname,Telephone
FROM acc_count, depositor
WHERE acc_count.IDNO =  depositor.IDNO and SumBalance < 5000
GO
```

单击工具栏上的“!”按钮，执行 T-SQL 语句，视图创建成功。

分析以上各视图是否为可更新视图，即能否通过此视图修改基本表中数据。由读者自行完成。

6.1.4 使用视图

视图创建完毕，就可以如同查询基本表一样通过视图查询所需要的数据，而且有些查询需求的数据直接从视图中获取比从基本表中获取更简单。此外，也可以通过视图修改基本表中的数据。

1. 使用视图进行数据查询

可以在 SSMS 中选中要查询的视图并打开，浏览该视图的数据；也可以在查询窗口中执行 T-SQL 语句查询视图。

例如，要查询各个储户的统计信息，就可以在 SSMS 中右击 acc_count 视图，选择“选择前 1000 行”命令或“编辑前 200 行”命令，即可浏览每个储户的账户个数和总存款余额，如图 6-9 所示。



图 6-9　界面方式查询视图中的数据

也可以在查询窗口中执行如下 T-SQL 语句：

```
SELECT * FROM acc_count
```

同样可以查询各储户的统计信息，如图 6-10 所示。



图 6-10　T-SQL 语句查询视图中的数据

【例 6-5】 在查询窗口中查询 s_c_sc 视图，统计“C++语言”课程的总分和平均分。

```
USE teaching
```

```
SELECT sumscore = SUM(score), avgscore = AVG(score) FROM s_c_sc
WHERE cname = 'C++语言'
```

本例的执行结果如图 6-11 所示。

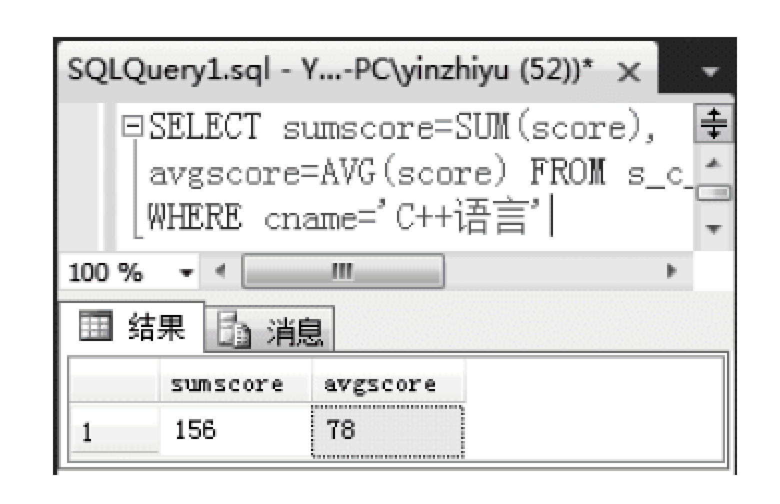


图 6-11 例 6-5 执行结果

【例 6-6】 查询 acc_count 视图中储户"李华"的统计信息。

```
USE bankcard
SELECT * FROM acc_count
WHERE Dname = '李华'
```

2. 使用视图修改基本表中的数据

修改视图的数据,其实就是对基本表中数据进行修改,因为真正存储数据的地方是基本表,而不是视图。修改视图的数据同样使用 INSERT、UPDATE、DELETE 语句来完成,但是在利用视图更新数据的时候也要注意一些事项,并不是所有的视图都可以进行数据更新,只有满足以下可更新条件的视图才能进行数据更新。

(1) 任何通过视图的数据更新(包括 UPDATE、INSERT 和 DELETE 语句)都只能引用一个基本表的列。

① 如果视图中数据为一个表的行、列子集,则此视图可更新(包括 UPDATE、INSERT 和 DELETE 语句);但如果视图中没有包含表中某个不允许取空值又没有默认值约束的列,则不能利用视图插入数据。

② 如果视图所依赖的基本表有多个,那么不能向该视图添加(INSERT)数据。

③ 如果视图所依赖的基本表有多个,那么一次修改只能修改(UPDATE)一个基本表中的数据。

④ 如果视图所依赖的基本表有多个,那么不能通过视图删除(DELETE)数据。

(2) 视图中被修改的列必须直接引用表列中的基础数据,不能是通过任何其他方式对表中的列进行派生而来的数据,比如通过聚合函数、计算(如表达式计算)、集合运算等。

(3) 被修改的列不应是在创建视图时受 GROUP BY、HAVING、DISTINCT 或 TOP 子句影响的。

注意:*通常有可能插入并不满足视图查询的 WHERE 子句条件的行。为了限制此类操作,可以在创建视图时使用 WITH CHECK OPTION 选项。*

【例 6-7】 通过 male_View 视图向 student 表中插入一个男生。

```
INSERT INTO male_View VALUES ('201501005', '张三', '男', '1997-8-1','2015-9-6','电子信息',
'2015 级')
```

如果通过 male_View 视图向 student 表中插入一个女生,也可以完成。如果不希望用

户通过 male_View 视图插入女生，那么在创建 male_View 视图时应该使用 WITH CHECK OPTION 选项。命令格式如下：

```
CREATE VIEW male_View   AS
SELECT sno, sname, ssex, sbirthday, en_time, specialty, grade
FROM student WHERE ssex = '男' WITH CHECK OPTION
```

6.1.5 删除视图

在不需要某个视图或想清除视图定义及与之相关联的权限时，可以删除该视图。视图的删除不会影响所依附的基本表的数据，定义在系统表 sysobjects、syscolumns、syscomments、sysdepends 和 sysprotects 中的视图信息也会被删除。

1. 在 SSMS 中删除视图

在 SSMS 中选中要删除的视图，右击选择“删除”命令，如图 6-12 所示，进入“删除对象”窗口，单击“确定”按钮就可以删除视图。



图 6-12 删除视图菜单

2. 使用 T-SQL 语句删除视图

T-SQL 提供了视图删除语句 DROP VIEW。

其语法格式如下：

```
DROP VIEW view_name
```

【例 6-8】 删除例 6-1 创建的 s_c_sc 视图。

```
USE teaching
```

```
GO
DROP VIEW s_c_sc
GO
```

6.2 索　　引

索引(Index)是对数据库表中一个或多个列的值进行排序的结构,其主要目的是提高SQL Server系统的性能,加快数据的查询速度和减少系统的响应时间。所以,索引就是加快检索表中数据的方法。

6.2.1 索引简介

数据库的索引就类似于书籍的目录,如果想快速查找而不是逐页查找指定的内容,可以通过目录中章节的页号快速找到其对应的内容。当表中有大量记录时,若要对表进行查询,第一种搜索信息方式是全表搜索,即将所有记录一一取出,和查询条件进行一一对比,然后返回满足条件的记录,这样做会消耗大量数据库系统时间,并造成大量磁盘I/O操作;第二种就是在表中建立索引,然后在索引中找到符合查询条件的索引值,最后通过保存在索引中的ROWID(行号,相当于页码)快速找到表中对应的记录。

索引包含从表或视图中一个或多个列生成的键,以及映射到指定数据的存储位置的指针,它是以B+树结构与表或视图相关联的。

索引的优点包括:

(1) 大大加快数据的检索速度,这是创建索引的最主要的原因。

(2) 创建唯一性索引,可以保证表中每一行数据的唯一性。

(3) 可以加速表和表之间的连接。

(4) 在使用分组和排序子句进行数据检索时,同样可以显著减少查询中分组和排序的时间。

(5) 查询优化器可以提高系统的性能,但它是依靠索引起作用的。

虽然索引具有如此多的优点,但索引的存在也让系统付出了一定的代价。创建索引和维护索引都会消耗时间,当对表中的数据进行增加、删除和修改操作时,索引就要进行维护,否则索引的作用就会降低。

另外,每个索引都会占用一定的物理空间,如果占用的物理空间过多,就会影响到整个SQL Server系统的性能。

6.2.2 索引类型

SQL Server 2008支持在表中任何列(包括计算列)上定义索引。索引可以是唯一的,即索引列不会有两行记录相同,这样的索引称为唯一索引。例如,如果在student表中的sname列上创建了唯一索引,则以后输入的姓名将不能同名。索引也可以是不唯一的,即索引列上可以有多行记录相同。如果索引是根据单列创建的,这样的索引称为单列索引,而根据多列组合创建的索引则称为复合索引。

按索引的组织方式的区分,可以将索引分为聚集索引和非聚集索引。

1. 聚集索引

聚集索引会对表和视图进行物理排序，所以这种索引对查询非常有效，在表和视图中只能有一个聚集索引。当建立主码约束时，如果表中没有聚集索引，SQL Server 会用主码列作为聚集索引键。可以在表的任何列或列的组合上建立索引，实际应用中一般为定义成主码约束的列建立聚集索引。

例如，汉语字典的正文就是一个聚集索引的顺序结构。

比如，要查“安”字，就可以翻开字典的前几页，因为“安”的拼音是 an，而按拼音排序的字典是以字母 a 开头以 z 结尾的，那么“安”字就自然地排在字典的前部。如果翻完了所有 an 读音的部分仍然找不到这个字，就说明字典中没有这个字。

同样，如果查“张”字，可以将字典翻到最后部分，因为“张”的拼音是 zhang。

也就是说，字典的正文内容本身就是按照音序排列的，而“汉语拼音音节索引”就可以称为“聚集索引”。

2. 非聚集索引

非聚集索引不会对表和视图进行物理排序，如果表中不存在聚集索引，则表是未排序的。在表或视图中，最多可以建立 250 个非聚集索引，或者 249 个非聚集索引和 1 个聚集索引。

例如，查字典时，不认识的字，就不能按照上面的方法来查找。

此时可以根据“偏旁部首”来查(以下内容因所使用字典不同而异)。比如查“张”字，在查部首之后的检字表中“张”的页码是 622 页，检字表中“张”的上面是“弛”字，但页码却是 60 页，“张”的下面是“弟”字，页码是 95 页，正文中这些字并不是真的分别位于“张”字的上下方。

所以，现在看到的连续的“弛、张、弟”三字实际上就是它们在非聚集索引中的排序，是字典正文中的字在非聚集索引中的映射。通过这种方式来找到所需要的字要两个过程，先找到目录中的结果，然后再翻到所需要的页码。

这种目录纯粹是目录、正文纯粹是正文的排序方式就称为“非聚集索引”。

聚集索引和非聚集索引都可以是唯一的索引。因此，只要列中数据是唯一的，就可在同一个表上创建一个唯一的聚集索引。如果必须实施唯一性以确保数据的完整性，则应在列上创建 UNIQUE 或 PRIMARY KEY 约束，而不要创建唯一索引。

创建 PRIMARY KEY 或 UNIQUE 约束会在表中指定的列上自动创建唯一索引。创建 UNIQUE 约束与手动创建唯一索引没有明显的区别，进行数据查询时，查询方式相同，而且查询优化器不区分唯一索引是由约束创建还是手动创建的。如果存在重复的键值，则无法创建唯一索引和 PRIMARY KEY 或 UNIQUE 约束。如果是复合的唯一索引，则该索引可以确保索引列中每个组合都是唯一的。创建复合唯一索引可为查询优化器提供附加信息，所以对多列创建复合索引时最好是唯一索引。

6.2.3 创建索引

我们已经知道，创建索引虽然可以提高查询速度，但是它需要牺牲一定的系统性能。因此，在创建时，哪些列适合创建索引，哪些列不适合创建索引，需要进行详细的考察。

1. 创建索引时应考虑的问题

(1) 对一个表创建大量的索引时,应进行权衡。

对于 SELECT 查询,大量索引可以提高性能,可以从中选择最快的查询方法;但是这会影响 INSERT、UPDATE 和 DELETE 语句的性能,因为对表中的数据进行修改时,对索引也要动态地维护,维护索引耗费的时间会随着数据量的增加而增加,所以应避免对经常更新的表建立过多的索引。而对更新少而且数据量大的表创建多个索引,则可以大大提高查询性能。

(2) 对于小型表(行数较少)进行索引可能不会产生优化效果。

(3) 对于主码和外键列应考虑建立索引,因为经常会通过主码查询数据,而外键用于表间的连接。

(4) 很少在查询中使用的列以及值很少的列(比如"性别"列)不应考虑建索引。

(5) 视图中如果包含聚合函数或连接时,创建视图的索引可以显著提升查询性能。

2. 通过 SSMS 创建索引

在 SSMS 中使用向导创建索引是一种图形界面环境下最快捷的创建方式,其步骤如下。

(1) 在 SSMS 的"对象资源管理器"面板中,选择要创建索引的表(如 teaching 数据库中的 student 表),然后展开 student 表前面的"+"号,选中"索引"选项,右击,在弹出的快捷菜单中选择"新建索引"命令,如图 6-13 所示。



图 6-13 新建索引菜单

(2) 选择"新建索引"命令后,进入"新建索引"窗口。在"常规"选择页中,可以创建索引。在"索引名称"文本框中输入索引名称,在"索引类型"下拉列表框中选择是不是聚集索引,通过"唯一"复选框可选择是否设置唯一索引等。例如,输入索引名称为 index_sname,选择"非聚集"选项,如图 6-14 所示。

(3) 通过索引设置按钮,可以为新建的索引添加、删除、移动索引列。例如,单击"添加"按钮,进入添加索引列窗口,选中 sname 列前的复选框,单击"确定"按钮即可添加一个按 sname 列升序排序的非聚集索引。再单击"确定"按钮,索引创建完成。

索引创建完成后,在 SSMS 的"对象资源管理器"面板中,选择创建了索引的表(student),展开 student 表前面的"+"号,再展开"索引"选项前面的"+"号,就会出现新建的索引 index_sname,如图 6-15 所示。

3. 利用 T-SQL 语句创建索引

使用 T-SQL 语句创建索引的语法格式如下:

```
CREATE [ UNIQUE ][ CLUSTERED | NONCLUSTERED ] INDEX index_name
  ON { table_name | view_name }  ( column_name [ ASC | DESC ] [ ,...n ] )
  [ WITH < index_option > [ ,...n] ] [ ON filegroup ]
< index_option > : : =
  { PAD_INDEX  |  FILLFACTOR = fillfactor
| IGNORE_DUP_KEY  |  DROP_EXISTING
| STATISTICS_NORECOMPUTE }
```

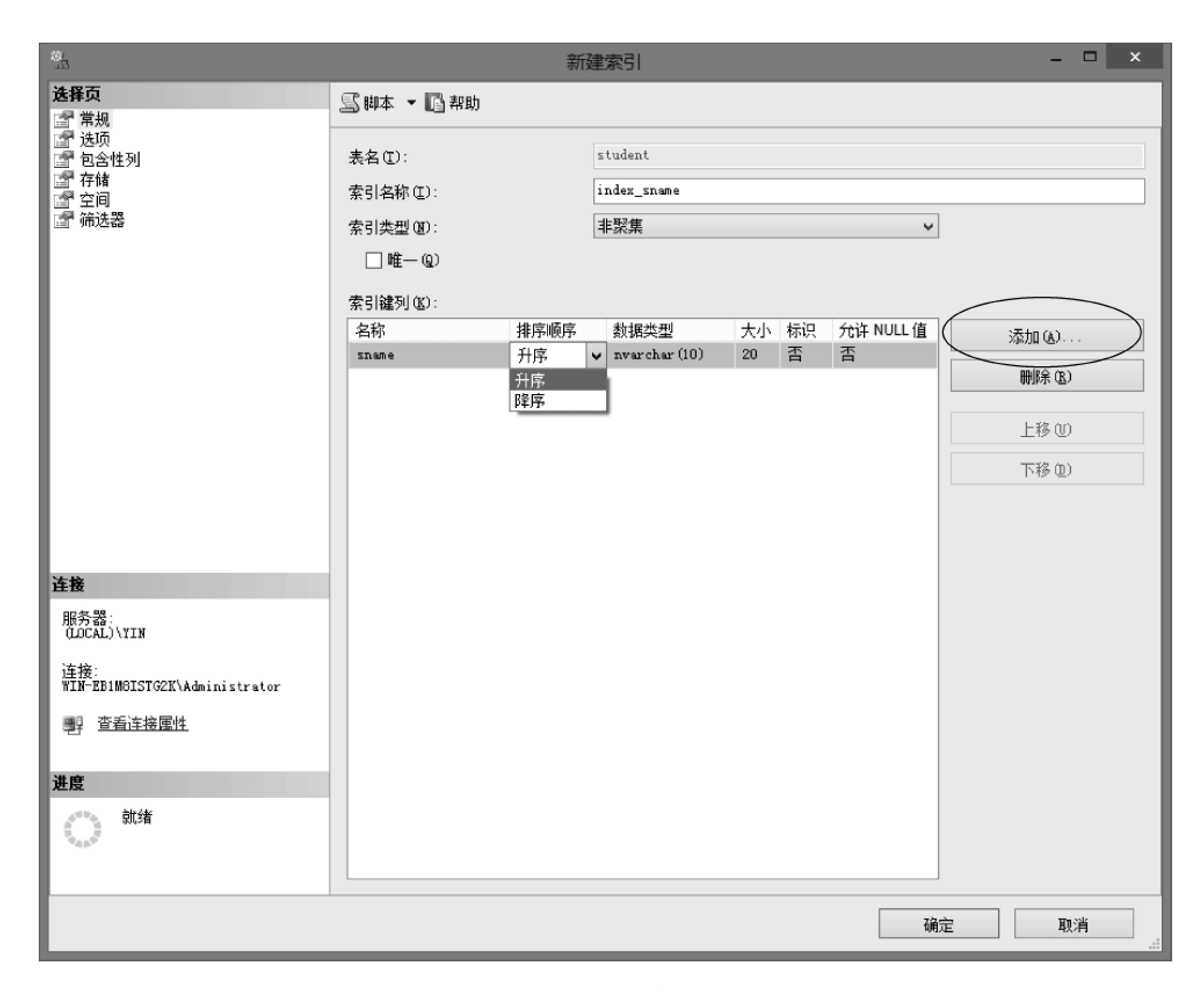


图 6-14　添加索引列



图 6-15　创建成功的索引

其中各参数说明如下。

(1) UNIQUE——建立的索引字段中不能有重复数据出现，即创建的索引是唯一索引。如果不使用这个关键字，创建的索引就不是唯一索引。

(2) CLUSTERED|NONCLUSTERED——指定 CLUSTERED 来建立聚集索引，使用 NONCLUSTERED 来创建非聚集索引，两者只能选其一。如果不指定，默认为非聚集索引。

(3) index_name——为新创建的索引指定的名字，索引名必须符合标识符的规则。

(4) table_name | view_name——创建索引的表或视图的名字。

(5) column_name——索引中包含的列的名字。

(6) FILLFACTOR——索引页的填充率,指定每个索引页预留多少可利用空间,可利用 WITH FILLFACTOR 语句指定其大小。如果要查询 FILLFACTOR 的大小,可以使用 SELECT index name,origfillfactor FROM sysindexes 命令进行查询。

(7) PAD_INDEX 和 FILLFACTOR——PAD_INDEX 只有在指定了 FILLFACTOR 时才能使用,属于填充因子。

(8) IGNORE_DUP_KEY——指在使用 INSERT 或 UPDATE 命令修改数据且加入相同关键字内容时对操作的反应。

(9) DROP_EXISTING——删除并重新建立原来存在的聚集索引或非聚集索引,新指定的索引名必须与现有的索引名相同。

(10) STATISTICS_NORECOMPUTE——过期的索引统计不会自动重新计算。

(11) filegroup——在已经创建的文件组上指定索引。

【例 6-9】 同前例,根据 teaching 数据库中 student 表的姓名列的升序创建一个名为 index_sname 的普通索引,用 T-SQL 语句完成。

```
USE teaching
GO
CREATE INDEX index_sname ON student(sname)
```

【例 6-10】 根据 teaching 数据库中 student 表的专业、年级创建一个名为 specialty_grade 的复合索引,其中专业名称为升序,年级为降序。

```
USE teaching
GO
CREATE INDEX specialty_grade
ON student(specialty ASC, grade DESC)
```

同样,索引创建完成后,在 SSMS 的"对象资源管理器"面板中,选择创建了索引的表(student),展开表前面的"+"号,再展开"索引"选项前面的"+"号,就会出现新建的索引 specialty_grade。

4. 间接创建索引

在定义表结构或修改表结构时,如果定义了主码约束(PRIMARY KEY)或者唯一性约束(UNIQUE),可以间接创建索引。

【例 6-11】 创建一个 student1 表,并定义主码约束。

```
USE teaching
GO
CREATE TABLE student1(
  sno char(6) PRIMARY KEY,
  sname char(8)   )
```

此例中,就按 sno 升序创建了一个聚集索引。

【例 6-12】 创建一个 teacher 教师表,并定义主码约束和唯一性约束。

```
USE teaching
GO
```

```
CREATE TABLE teacher(
  tno char(6) PRIMARY KEY,
  tid char(18) UNIQUE,
  tname nchar(4) )
```


此例中创建了两个索引：按 tno(教师号)升序创建了一个聚集索引，按 tid(身份证号)升序创建了一个非聚集唯一索引。

索引一经创建，就完全由系统自动选择和维护，不需要用户指定使用索引，也不需要用户执行打开索引或进行重新索引等操作，所有的工作都由 SQL Server 数据库管理系统自动完成。但对于读者来讲，应该明白为什么要创建这些索引，即这些索引可能在什么情况下被选择使用。例如，student 表中按姓名列升序创建的 index_sname 索引，下面的 T-SQL 语句在执行时系统就可以利用此索引来加快查询速度。

(1) SELECT sno，specialty FROM student WHERE sname='郑丽'

(2) DELETE FROM student WHERE sname='郑丽'

5. 创建视图的索引

视图也称为虚拟表，这是因为由视图返回的结果集的一般格式与由列和行组成的表相似，并且，在 SQL 语句中引用视图的方式也与引用表的方式相同。

对于标准视图而言，结果集不是永久地存储在数据库中，为每个引用视图的查询动态生成结果集的开销很大，特别是对于那些涉及对大量行进行复杂处理(如聚合大量数据或连接许多行)的视图，其开销更为可观。若经常在查询中引用这类视图，可通过在视图上创建索引来提高性能。在视图上创建唯一聚集索引时将执行该视图，并且结果集在数据库中的存储方式与带聚集索引的表的存储方式相同。

在视图上创建索引的另一个好处是：查询优化器开始在查询中使用视图的索引，而不是直接在 FROM 子句中命名视图。这样一来，可从视图的索引检索数据而无须重新编码，由此带来的高效率也使现有查询获益。

在视图上创建聚集索引可存储创建索引时存在的数据。视图的索引还自动反映自创建索引后对基本表数据所做的更改，这一点与在基本表上创建的索引相同。视图的聚集索引必须唯一，从而提高了 SQL Server 在索引中查找数据行的效率。

与基本表上的索引相比，对视图中索引的维护可能更复杂。只有当视图的结果检索速度的效益超过了修改所需的开销时，才应在视图上创建索引。这样的视图通常包括映射到相对静态的数据上、处理多行以及由许多查询引用的视图。

在视图上创建聚集索引之前，该视图必须满足下列要求：

(1) 当执行 CREATE VIEW 语句时，ANSI_NULLS 和 QUOTED_IDENTIFIER 选项必须设置为 ON。OBJECTPROPERTY 函数通过 ExecIsAnsiNullsOn 或 ExecIsQuotedIdentOn 属性为视图报告此信息。

(2) 为执行所有 CREATE TABLE 语句以创建视图引用的表，ANSI_NULLS 选项必须设置为 ON。

(3) 视图不能引用任何其他视图，只能引用基本表。

(4) 视图引用的所有基本表必须与视图位于同一个数据库中，并且所有者也与视图相同。

(5) 必须使用 SCHEMABINDING 选项创建视图。SCHEMABINDING 将视图绑定到基础基本表的架构上。

(6) 必须已使用 SCHEMABINDING 选项创建了视图中引用的用户定义的函数。

(7) 表和用户定义的函数必须由两部分的名称引用。

(8) 视图中的表达式所引用的所有函数必须是确定性的。OBJECTPROPERTY 函数的 IsDeterministic 属性报告用户定义的函数是否是确定性的。

(9) 选择列表不能使用 * 或 table_name. * 语法指定列,必须显式给出列名。

(10) 不能在多个视图列中指定用作简单表达式的表的列名。如果对列的所有(或只有一个例外)引用是复杂表达式的一部分或是函数的一个参数,则可多次引用该列。

在视图上创建的第一个索引必须是唯一聚集索引,在创建唯一聚集索引后,可创建其他非聚集索引。视图上的索引命名规则与表上的索引命名规则相同。

可以在 SSMS 的对象资源管理器中以界面方式创建视图的索引,也可以使用 T-SQL 语句创建视图的索引。

【例 6-13】 创建一个 female_view 女生视图,并为该视图按 sno 升序创建一个具有唯一性的聚集索引。

创建视图:

```
USE teaching
GO
CREATE VIEW female_view
WITH SCHEMABINDING
AS
SELECT sno,sname,ssex,specialty FROM dbo.student
WHERE ssex = '女'
```

在 SSMS 的对象资源管理器中,展开相应的数据库文件夹,展开"视图"选项,再展开要创建索引的视图。例如,要在 female_view 视图创建聚集索引,则应展开该视图,然后右击"索引"选项,选择"新建索引"命令,如图 6-16 所示。在"新建索引"对话框中按学号的升序创建一个唯一的聚集索引,输入索引名,设置索引类型,单击"确定"按钮即可新建一个视图的索引。

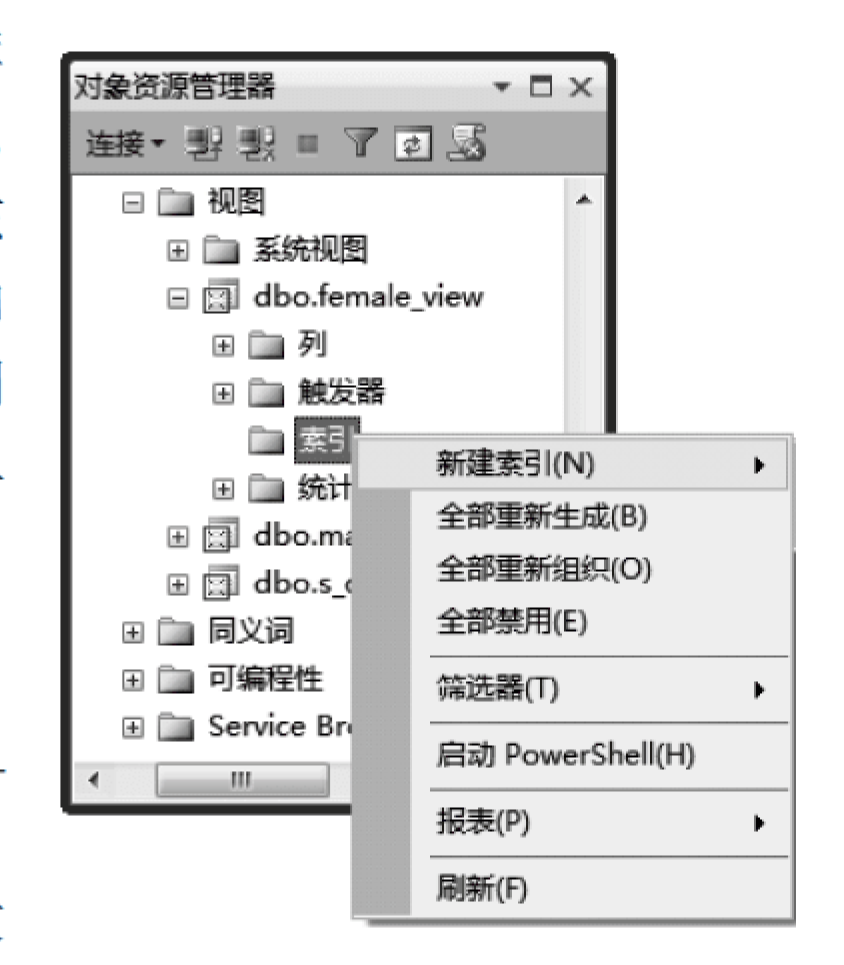


图 6-16 新建索引视图

T-SQL 语句创建索引:

```
CREATE UNIQUE CLUSTERED INDEX index_female ON female_
view(sno)
```

创建聚集索引后,对于任何试图对视图修改基本数据而进行的连接,其选项设置必须与创建索引所需的选项设置相同。如果这个执行语句的连接没有适当的选项设置,则 SQL Server 会生成错误并回滚任何会影响视图结果集的 INSERT、UPDATE 或 DELETE 语句。有关更多信息,请参见 SQL Server 联机丛书中影响结果的 SET 选项。

若删除视图,视图上的所有索引也将被删除。若删除聚集索引,视图上的所有非聚集索

引也将被删除。可分别除去非聚集索引。除去视图上的聚集索引将删除存储的结果集，并且优化器将重新像处理标准视图那样处理视图。

尽管 CREATE UNIQUE CLUSTERED INDEX 语句仅指定组成聚集索引键的列，但视图的完整结果集将存储在数据库中。与基本表上的聚集索引一样，聚集索引的 B+树结构仅包含键列，但数据行包含视图结果集中的所有列。

若想为现有系统中的视图添加索引，必须计划绑定任何想要放入索引的视图。可以除去视图并通过指定 WITH SCHEMABINDING 重新创建它；也可以创建另一个视图，使其具有与现有视图相同的文本，但是名称不同。优化器将考虑新视图上的索引，即使在查询的 FROM 子句中没有直接引用它。

6.2.4 查看索引信息

在实际使用索引的过程中，有时需要对表的索引信息进行查询，了解在表中曾经建立的索引。可以使用 SSMS 进行查询；也可以在查询窗口中使用 T-SQL 语句进行查询。

1. 在 SSMS 中查看索引信息

在 SSMS 中，选择要查看的表，然后右击相应的表，从菜单中选择"设计"命令，进入"表设计器"窗口，右击任意位置，选择"索引/键"命令即可查看此表上所有的索引信息。例如，查看 student 表上的索引信息，如图 6-17 和图 6-18 所示。



图 6-17 查看 student 表上的索引信息

2. 使用 T-SQL 语句查看索引信息

可以使用系统存储过程 sp_helpindex 或 sp_help 来查看索引信息，以下以查看 student 表上的索引信息为例。

(1) 使用系统存储过程 sp_helpindex 查看索引信息。

```
USE teaching
GO
EXEC sp_helpindex student
```

执行结果如图 6-19 所示。



图 6-18　student 表上的索引



图 6-19　sp_helpindex 查看 student 表上的索引

结果显示了 student 表中所建立的三个索引。

① 索引名称为 index_sname，索引描述为非聚集索引，索引关键字为 sname。

② 索引名称为 PK_student，索引描述为聚集索引、唯一索引，索引关键字为 sno。

③ 索引名称为 specialty_grade，索引描述为非聚集索引，索引关键字为 specialty 和 grade。

（2）使用系统存储过程 sp_help 查看索引信息。

```
USE teaching
GO
EXEC sp_help student
```

执行结果如图 6-20 所示。

由结果可以看出，执行 sp_help 系统存储过程查询的结果要比执行 sp_helpindex 显示的结果更加详细，除了索引信息，还包括当前表的基本信息、与此表相关的各种约束等。



图 6-20 sp_help 查看 student 上的索引

6.2.5 删除索引

当不再需要一个索引时，可以将其从数据库中删除，以释放当前占用的存储空间，这些释放的空间可以由数据库中的任何对象使用。

删除聚集索引可能要花费一些时间，因为必须重建同一个表上的所有非聚集索引。另外，必须通过删除约束才能删除 PRIMARY KEY 或 UNIQUE 约束使用的索引。如果要在不删除和重新创建 PRIMARY KEY 或 UNIQUE 约束的情况下，删除并重新创建该约束使用的索引，则应该通过一个步骤重建该索引。删除某个表时，会自动删除在此表上创建的索引。

1. 在 SSMS 中删除索引

与在 SSMS 中创建索引的步骤一样，选中要删除索引的表，选中“索引”选项，展开“索引”选项前面的“+”号，如图 6-21 所示，右击要删除的索引，选择“删除”命令，弹出“删除对象”对话框，单击“确定”按钮。

2. 使用 T-SQL 语句删除索引

删除索引的 T-SQL 语句的语法格式为：

```
DROP  INDEX  table_name.index_name
```

【例 6-14】 删除 student 表中的 index_sname 索引。

```
DROP  INDEX  student.index_sname
GO
```



图 6-21　删除索引菜单

习　题　6

1. 引入视图的主要目的是什么?

2. 当删除基本表时,基于它创建的视图会被删除吗?

3. 简述视图的优点。

4. 可更新视图必须满足哪些条件?

5. 创建索引的必要性和作用是什么?

6. 简述索引的优点。

7. 聚集索引和非聚集索引有何异同?

8. 在 SSMS 中以界面方式创建一个名为 depositor_AccNO 的视图,对储户表和账户表连接查询,要求包含储户基本信息和他们拥有的卡号。然后,分析此视图是否为可更新视图,说明理由。

9. 在 SSMS 中以界面方式创建一个包含计算机专业学生基本信息的视图 computer_stu。然后,分析此视图是否为可更新视图,说明理由。

10. 利用 T-SQL 语句创建一个每个学生的平均成绩的视图 s_avgscore,要求包含学生的学号和姓名。然后,分析此视图是否为可更新视图,说明理由。

11. 利用 T-SQL 语句创建一个每个年级、每个专业各门课平均成绩的视图 avgscore。然后,分析此视图是否为可更新视图,说明理由。

12. 在 SSMS 中以界面方式按照 sc 表的成绩列升序创建一个普通索引(非唯一、非聚集),并举例说明什么查询语句会利用此索引加快查询速度。

13. 利用 T-SQL 语句按照 account 表的余额列降序创建一个普通索引,并举例说明什么查询语句会利用此索引加快查询速度。

14. 利用 T-SQL 语句按照 Trecord 表的账号列升序创建一个普通索引,并举例说明什么查询语句会利用此索引加快查询速度。

第7章 存储过程和触发器

在 SQL Server 2008 应用操作中，存储过程和触发器都扮演着相当重要的角色。

存储过程可以使用户对数据库的管理工作变得更容易。当开发一个应用程序时，为了易于修改和扩充，经常会将负责不同功能的语句集中起来而且按照用途分别独立放置，以便能够反复调用，这些独立放置且拥有不同功能的语句，就是“过程”(Procedure)。

触发器是一种特殊类型的存储过程。当有操作影响到触发器保护的数据时，触发器就会自动触发执行。触发器是与表紧密联系在一起的，它在特定的表上定义，并与指定的数据修改事件相对应；它是一种功能强大的工具，可以扩展 SQL Server 完整性约束默认值对象和规则的完整性检查逻辑，实施更为复杂的数据完整性约束。

本章主要介绍存储过程的基本概念，存储过程的创建、修改、调用和删除操作；触发器的基本概念，触发器的分类，触发器的创建、修改和删除，以及触发器的应用。

7.1 存储过程

SQL Server 2008 的存储过程(Stored Procedure)就是一个具有独立功能的子程序，以特定的名称存储在数据库中。可以在存储过程中声明变量、有条件地执行语句以及实现其他各项强大的程序设计功能。

7.1.1 存储过程概述

存储过程是 T-SQL 语句和可选流程控制语句的预编译集合，它以一个名称存储并作为一个单元处理，能够提高系统的应用效率和执行速度。SQL Server 提供了许多系统存储过程以管理 SQL Server 和显示有关数据库和用户的信息。

存储过程是一种独立存储在数据库内的对象，可以带输入参数、输出参数，返回单个或多个结果集以及返回值，由应用程序通过调用执行。存储过程可以由客户调用，也可以从另一个过程或触发器调用，参数可以被传递和返回，出错代码也可以被检验。

存储过程最主要的特色是当写完一个存储过程后，该过程即被翻译成可执行码存储在系统表内，当作是数据库的对象之一，一般用户只要执行存储过程，并且提供存储过程所需的参数就可以得到所要的结果而不必再去编辑 T-SQL 命令。

一般来讲，应使用 SQL Server 中的存储过程而不使用存储在客户计算机本地的 T-SQL 程序，其优势主要表现在：

(1) 模块化程序设计。只需创建一次并将其存储在数据库中，以后即可在程序中调用该过程任意多次。存储过程可由在数据库编程方面有专长的人员创建，并可独立于程序源

代码而单独修改。如果业务规则发生变化，可以通过修改存储过程来适应新的业务规则，而不必修改客户端的应用程序。这样所有调用该存储过程的应用程序就会遵循新的业务规则。

(2) 加快 T-SQL 语句的执行速度。如果某操作需要大量 T-SQL 语句或需重复执行，存储过程将比批处理代码的执行速度快。创建存储过程时对其进行分析和优化并预先编译好放在数据库内，减少编译语句所占用的时间；编译好的存储过程会进入缓存，所以对于经常执行的存储过程，除了第一次执行外，其他次执行的速度会有明显提高。而客户计算机本地的 T-SQL 语句每次运行时，都要从客户端重复发送，并且在 SQL Server 每次执行这些语句时，都要对其进行编译和优化。

(3) 减少网络流量。一个需要数百行 T-SQL 语句的操作由一条执行过程代码的单独语句就可实现，而不需要在网络中发送数百行代码。

(4) 更高的安全性。用户可以通过得到权限来执行存储过程，数据库不必给予用户直接访问数据库对象的权限，这些对象将由存储过程来执行操作。另外，存储过程可以加密，这样用户就无法阅读存储过程中的 T-SQL 语句。这些安全特性将数据库结构和数据库用户隔离开来，这也进一步保证了数据的完整性和可靠性。

7.1.2 存储过程的类型

1. 系统存储过程

存储过程在运行时生成执行方式，其后在运行时执行速度很快。SQL Server 2008 中的许多管理活动都是通过一种特殊的存储过程执行的，这种存储过程被称为系统存储过程。系统过程主要存储在 master 数据库中并以“sp_”为前缀，并且系统存储过程主要是从系统表中获取信息，从而为数据库系统管理员管理 SQL Server 提供支持。通过系统存储过程，SQL Server 中的许多管理性或信息性的活动(如获取数据库和数据库对象的信息)都可以顺利有效地完成。

尽管这些系统存储过程被存储在 master 数据库中，但是仍可以在其他数据库中对其进行调用，在调用时，不必在存储过程名前加上数据库名。而且当创建一个数据库时，一些系统存储过程会在新的数据库中被自动创建。

SQL Server 2008 系统存储过程是为用户提供方便的，它们使用户可以很容易地从系统表中提取信息、管理数据库，并执行涉及更新系统表的其他任务。

如果过程以“sp_”开始，又在当前数据库中找不到，SQL Server 2008 就在 master 数据库中寻找。当系统存储过程的参数是保留字或对象名，且对象名由数据库或拥有者名字限定时，整个名字必须包含在单引号中。一个用户需要在所有数据库中拥有执行一个系统存储过程的许可权，否则在任何数据库中都不能执行系统存储过程。

2. 本地存储过程

本地存储过程也就是用户自行创建并存储在用户数据库中的存储过程，一般所说的存储过程指的就是本地存储过程。

用户创建的存储过程是由用户创建并能完成某一特定功能(如查询用户所需的数据信息)的存储过程。

3. 临时存储过程

临时存储过程可分为以下两种：

1）本地临时存储过程

不论哪一个数据库是当前数据库，如果在创建存储过程时，其名称以“#”号开头，则该存储过程将成为一个存放在 tempdb 数据库中的本地临时存储过程。对于本地临时存储过程，只有创建它的连接的用户才能够对它操作，而且一旦这位用户断开与 SQL Server 的连接，本地临时存储过程就会自动删除。当然，这位用户也可以在连接期间用 DROP PROCEDURE 命令删除他所创建的本地临时存储过程。

2）全局临时存储过程

不论哪一个数据库是当前数据库，只要所创建的存储过程名称是以两个“#”号开头，则该存储过程将成为一个存储在 tempdb 数据库中的全局临时存储过程。全局临时存储过程一旦创建，以后连接到 SQL Server 2008 的任意用户都能执行它，而且不需要特定的权限。

当创建全局临时存储过程的用户断开与 SQL Server 2008 的连接时，SQL Server 2008 将检查是否有其他用户正在执行该全局临时存储过程，如果没有，便立即将该全局临时存储过程删除；如果有，SQL Server 2008 会让这些正在执行中的操作继续进行，但是不允许任何用户再执行该全局临时存储过程，等到所有未完成的操作执行完毕后，就会自动删除该全局临时存储过程。

不论创建的是本地临时存储过程还是全局临时存储过程，只要 SQL Server 2008 停止运行，它们将不复存在。

4. 远程存储过程

在 SQL Server 2008 中，远程存储过程是位于远程服务器上的存储过程，通常可以使用分布式查询和 EXECUTE 命令执行一个远程存储过程。

5. 扩展存储过程

扩展存储过程是用户可以使用外部程序语言（例如 C 语言）编写的存储过程。显而易见，扩展存储过程可以弥补 SQL Server 2008 的不足，并按需要自行扩展其功能。

扩展存储过程在使用和执行上与一般的存储过程完全相同，为了区别，扩展存储过程的名称通常以“XP_”开头。扩展存储过程以动态链接库（DLL）的形式存在，能让 SQL Server 2008 动态地装载和执行。扩展存储过程一定要存储在系统数据库 master 中。

7.1.3 创建存储过程

在 SQL Server 2008 中创建存储过程主要有两种方式：一种方式是在 SSMS 中以界面方式创建存储过程；另一种方式是通过在查询窗口中执行 T-SQL 语句创建存储过程。

1. 在 SSMS 中创建存储过程

在 SSMS 中以界面方式创建存储过程的步骤如下：

（1）打开 SSMS，展开要创建存储过程的数据库，展开“可编程性”选项，可以看到存储过程列表中系统自动为数据库创建的系统存储过程。右击“存储过程”选项，选择“新建存储过程”命令，如图 7-1 所示。

（2）出现创建存储过程的 T-SQL 命令，编辑相关的命令即可，如图 7-2 所示。

（3）命令编辑成功后，进行语法检查，然后单击“!”按钮，至此，一个新的存储过程建立



图 7-1　在 SSMS 中以界面方式创建存储过程

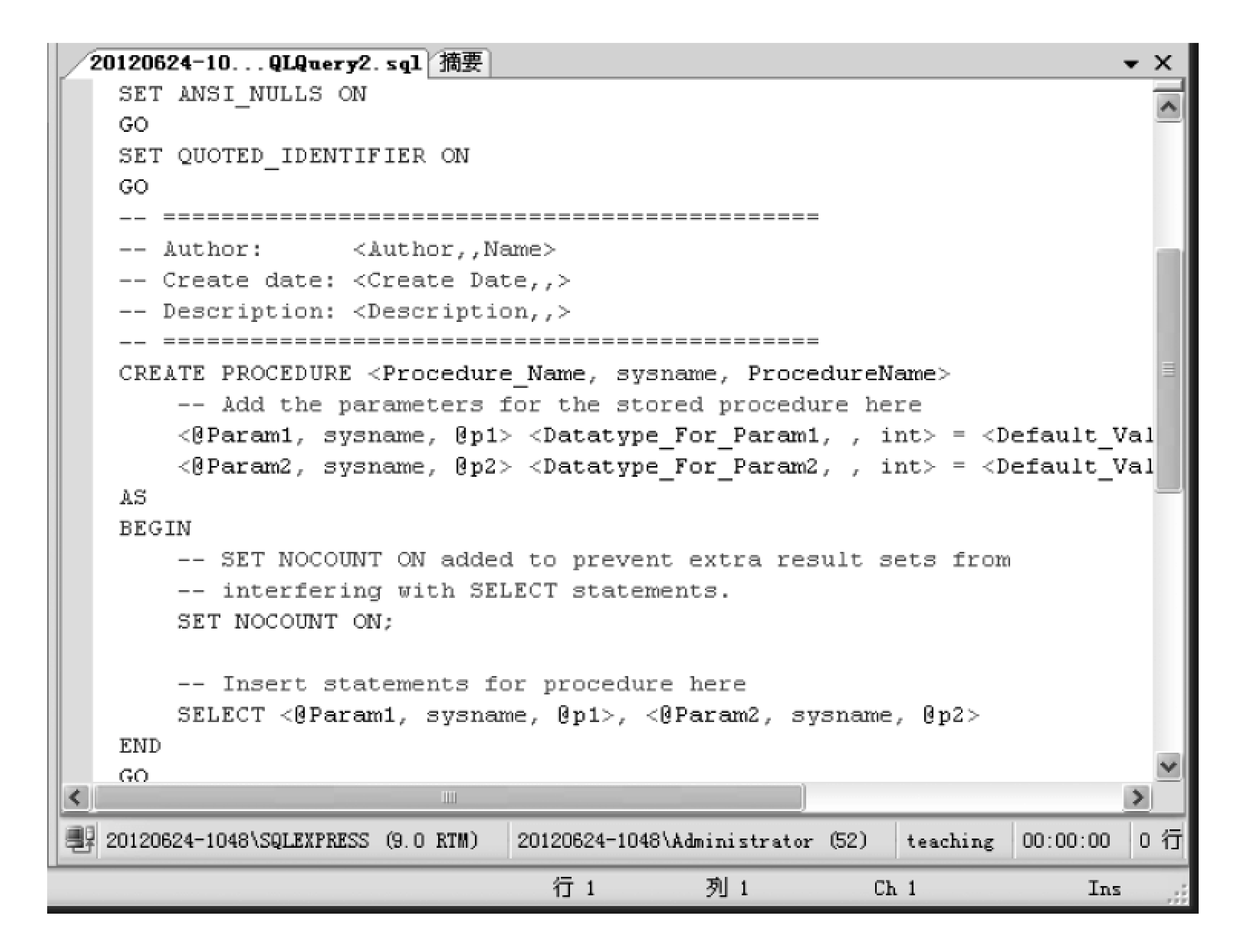


图 7-2　创建存储过程的 T-SQL 命令

成功。

注意：用户只能在当前数据库中创建存储过程，数据库的拥有者有默认的创建权限，权限也可以转让给其他用户。

2. 利用 T-SQL 语句创建存储过程

SQL Server 2008 提供了 CREATE PROCEDURE 语句来创建存储过程。

语法格式如下：

```
CREATE { PROC | PROCEDURE } procedure_name [ ; number ]
  [ { @parameter data_type }
  [ VARYING ] [ = default ] [ [ OUT [ PUT ] ] [ ,...n ]
```

```
[ WITH { RECOMPILE | ENCRYPTION  | RECOMPILE , ENCRYPTION } [ ,...n ] ]
[FOR REPLICATION]
AS sql_statement [ ...n ]
```


其中各参数说明如下：

(1) procedure_name——新建存储过程的名称。过程名必须符合标识符规则，且对于数据库及其所有者必须唯一。

(2) number——可选的整数，用来对同名的过程分组，以便用一条 DROP PROCEDURE 语句即可将同组的过程一起除去。

(3) @parameter——过程中的参数，在 CREATE PROCEDURE 语句中可以声明一个或多个参数。存储过程最多可以指定 2100 个参数，使用@符号作为第一个字符来指定参数名称，参数名称必须符合标识符的规则。

(4) data_type——参数的数据类型。所有数据类型(包括 text、ntext 和 image)均可以用作存储过程的参数。

(5) VARYING——指定作为输出参数支持的结果集。

(6) default——参数的默认值。如果定义了默认值，不必指定该参数的值即可执行过程。

(7) OUTPUT——表明参数是返回参数。该选项的值可以返回给调用此过程的应用程序。

(8) RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION——RECOMPILE 表明 SQL Server 不会缓存该过程的计划，该过程在运行时重新编译；ENCRYPTION 表示 SQL Server 加密用 CREATE PROCEDURE 语句创建的存储过程的定义，使用 ENCRYPTION 可防止将过程作为 SQL Server 复制的一部分发布。

(9) FOR REPLICATION——指定不能在订阅服务器上执行为复制创建的存储过程。使用 FOR REPLICATION 选项创建的存储过程可用于存储过程筛选，且只能在复制过程中执行。此选项不能和 WITH RECOMPILE 选项一起使用。

(10) AS——指定过程要执行的操作。

(11) sql_statement——过程要包含的任意数目和类型的 T-SQL 语句。

在创建存储过程时，应当注意以下几点。

(1) 存储过程最大不能超过 128MB。

(2) 用户定义的存储过程只能在当前数据库中创建，但是临时存储过程通常是在 tempdb 数据库中创建的。

(3) 在一条 T-SQL 语句中，CREATE PROCEDURE 不能与其他 T-SQL 语句一起使用。

(4) SQL Server 允许在存储过程创建时引用一个不存在的对象，在创建的时候，系统只检查创建存储过程的语法。存储过程在执行的时候，如果缓存中没有一个有效的计划，则会编译生成一个可执行计划。只有在编译的时候，才会检查存储过程所引用的对象是否都存在。这样，一个创建存储过程的语句只要在语法上没有错误，即使引用了不存在的对象也是可以成功执行的。但是，如果在执行(调用存储过程)的时候，存储过程引用了一个不存在的对象，这次执行操作将会失败。

【例 7-1】 在 teaching 数据库中创建无参存储过程，查询每个学生各门课程的平均成绩。

```
USE teaching
GO
CREATE PROCEDURE student_avg
AS
SELECT sno, avg(score)  as  'avgstore'  FROM  sc
GROUP BY sno
GO
```

命令执行，创建存储过程成功，如图 7-3 所示。

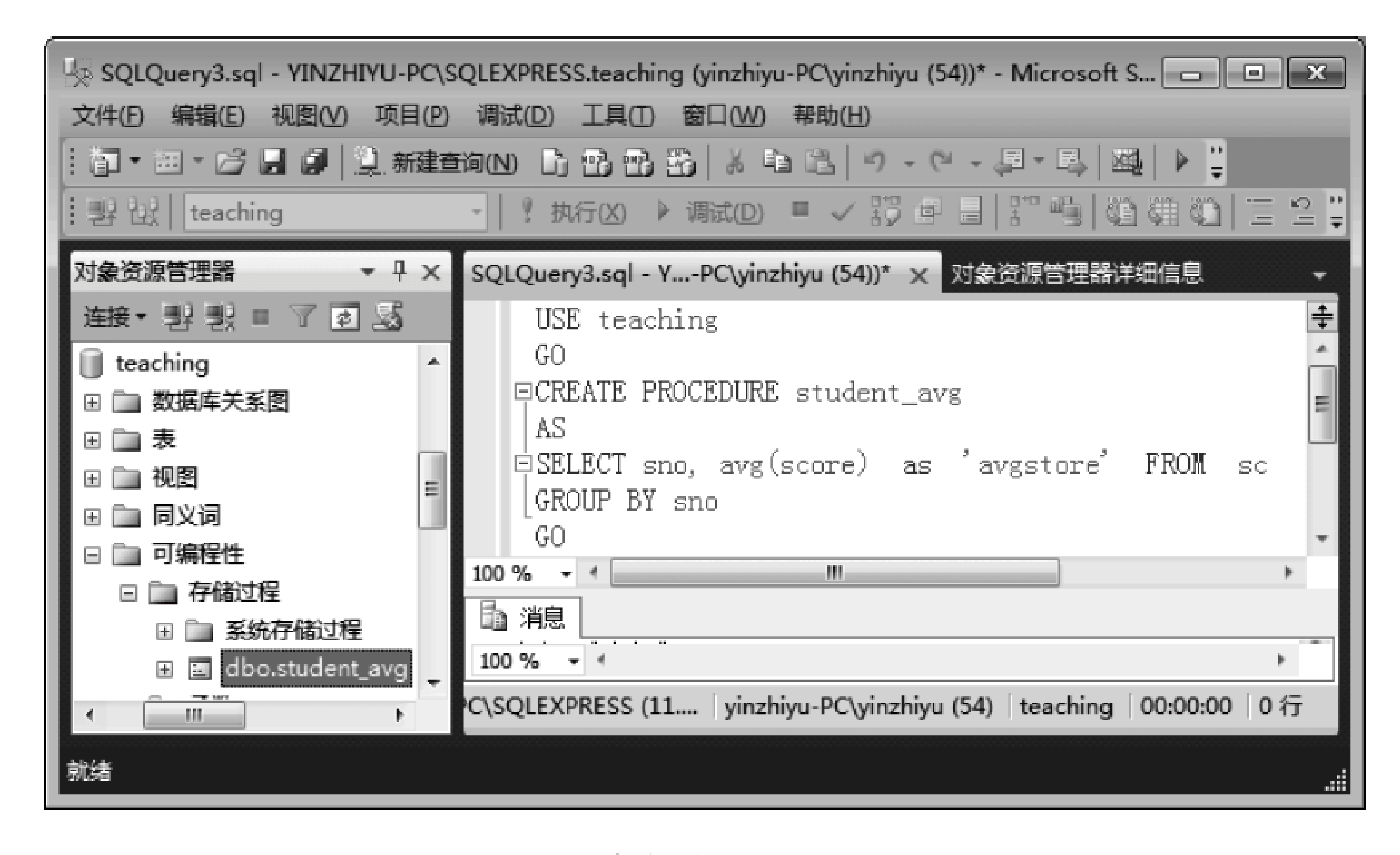


图 7-3 创建存储过程 student_avg

成功执行 CREATE PROCEDURE 语句后，创建的存储过程的名称存储在 sysobjects 系统表中，而 CREATE PROCEDURE 语句的文本存储在 syscomments 中。

【例 7-2】 在 teaching 数据库中创建带参数的存储过程，查询某个学生的基本信息。

```
USE teaching
GO
CREATE PROCEDURE GetStudent @number char(9)
AS
SELECT * FROM student WHERE sno = @number
GO
```

【例 7-3】 在 teaching 数据库中创建带参数的存储过程，修改某个学生某门课的成绩。

```
USE teaching
GO
CREATE PROCEDURE Update_score @number char(7),@cno char(4),@score int
AS  UPDATE sc SET score = @score
WHERE sno = @number and cno = @cno
```

【例 7-4】 在 bankcard 数据库中使用流程控制语句创建存储过程。假设今天银行有活动,如果今天某账号交易支出总金额超过 3000 元,则奖励其 10 元。

```
USE bankcard
GO
CREATE PROCEDURE add_10 @AccNO char(20)
WITH ENCRYPTION
AS
IF (SELECT SUM(Expense) FROM Trecord WHERE TDate = GETDATE() AND AccNO = @AccNO)> = 3000
BEGIN
  UPDATE Account SET Balance = Balance + 10 WHERE AccNO = @AccNO
  INSERT Trecord(TDate,AccNO,Income,Abstract) VALUES (GETDATE(),@AccNO,10, '银行活动奖励')
END
```

【例 7-5】 在 bankcard 数据库中创建带 OUTPUT 参数的存储过程,用于计算指定的储户的总余额。存储过程中使用一个输入参数(身份证号)和两个输出参数(储户姓名和总余额)。

```
USE bankcard
GO
CREATE PROCEDURE s _ balance @ IDNO char ( 18 ), @ dname nvarchar ( 10 ) OUTPUT,
@sbalancemoney OUTPUT
AS
SELECT @dname = Dname FROM depositor WHERE IDNO = @IDNO
SELECT @sbalance = SUM(Balance) FROM account WHERE IDNO = @IDNO
GO
```

7.1.4 执行存储过程

执行存储过程即调用存储过程,可以使用 SSMS 界面,也可以使用 T-SQL 语句中的 EXECUTE 命令。

1. 使用 SSMS 执行存储过程

在 SSMS 中以界面方式执行存储过程的步骤如下:

(1) 打开 SSMS,展开存储过程所在的数据库,展开“可编程性”选项,右击存储过程名,如 teaching 数据库中的 GetStudent,在弹出的快捷菜单中选择“执行存储过程”命令,如图 7-4 所示。

(2) 进入“执行过程”对话框,输入要查询的学生的 sno,如 201501001,如图 7-5 所示。

(3) 单击“确定”按钮,执行结果如图 7-6 所示。

2. 使用 T-SQL 语句执行存储过程

如果存储过程是批处理中的第一条语句,那么不使用 EXECUTE 关键字也可以执行该存储过程。对于存储过程的所有者或任何一名对此过程拥有 EXECUTE 权限的用户,都可以执行此存储过程。如果需要在启动 SQL Server 时,由系统自动执行存储过程,则可以使用 sp_procoption 进行设置。如果被调用的存储过程需要参数输入,可在存储过程名后逐一给定,每一个参数用逗号隔开,不必使用括号。如果没有使用“@参数名=default”这种方式



图 7-4 界面方式执行存储过程

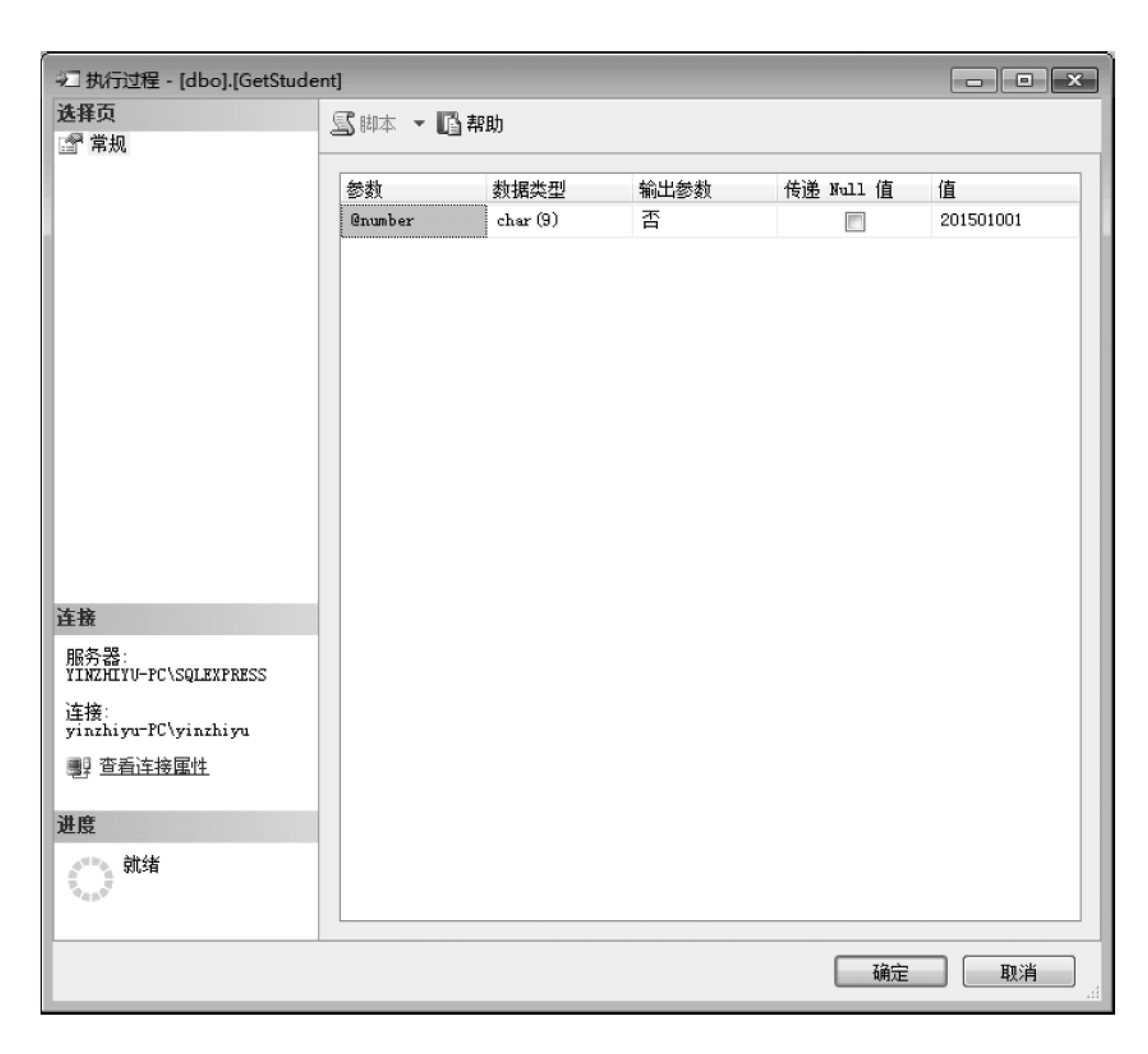


图 7-5 输入参数值

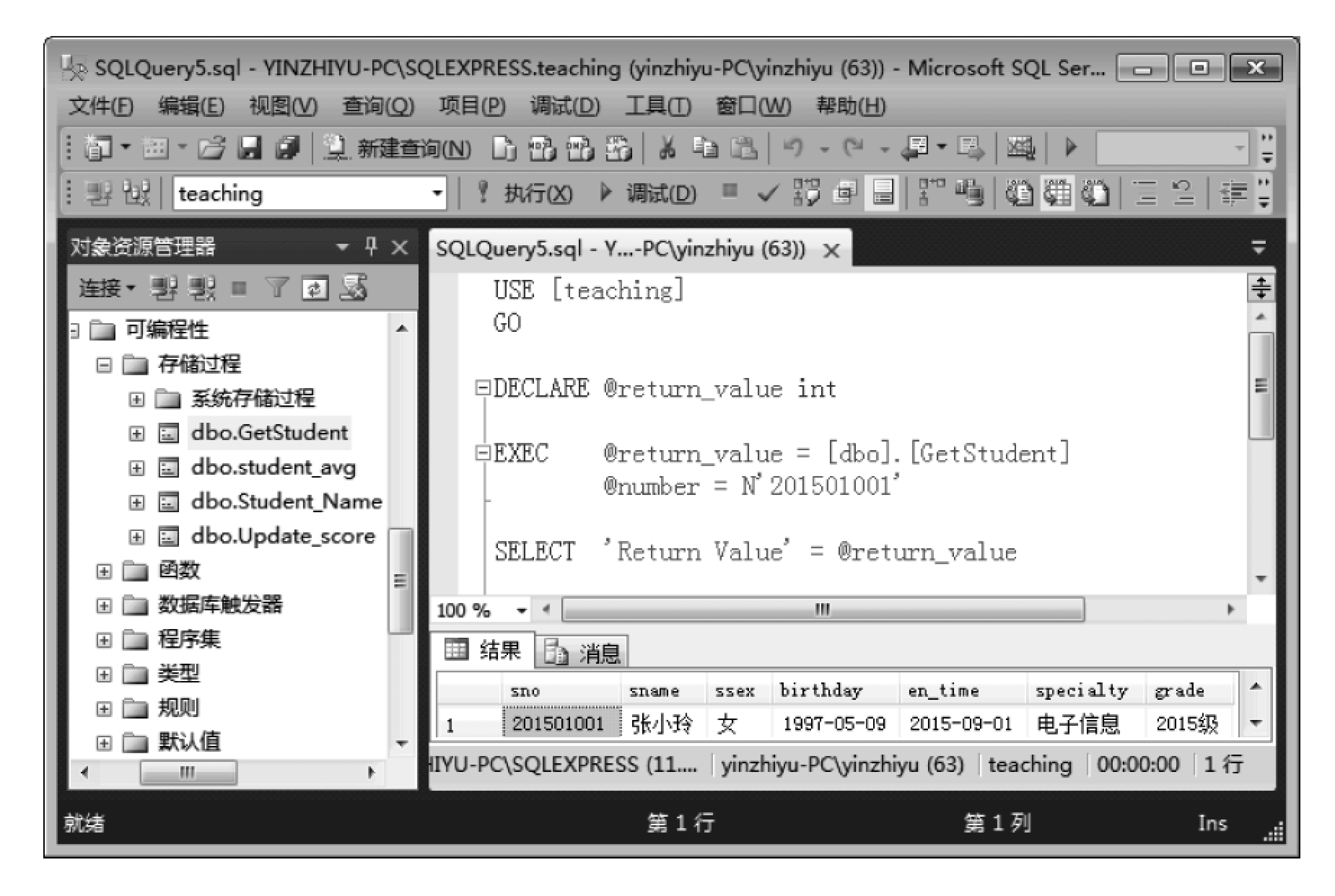


图 7-6　存储过程执行结果

传入值，则参数的排列必须和建立存储过程时所定义的次序对应，用来接收输出值的参数则必须加上 OUTPUT。

EXECUTE 可以简写为 EXEC，如果存储过程是批处理中的第一条语句，那么可以省略 EXECUTE 关键字。对于以 sp_开头的系统存储过程，系统将在 master 数据库中查找。如果执行用户自定义的 sp_开头的存储过程，就必须用数据库名和所有者名限定。

EXECUTE 语句的语法格式为：

```
[ [EXEC[UTE] ] [@return_status = ] procedure_name[;number]
{[[@parameter = ]value | [@ parameter = ] @variable [OUTPUT]]}
[WITH RECOMPILE ]
```

其中各参数说明如下：

（1）@return_status——一个可选的整型变量，保存存储过程的返回状态。

（2）procedure_name——执行的存储过程的名称。

（3）number——可选的整数，用于将相同名称的过程进行组合，使得它们可以用一条 DROP PROCEDURE 语句除去。

（4）@parameter——过程参数，在 CREATE PROCEDURE 语句中定义。参数名称前必须加上符号@。在以“@parameter = value”格式使用时，参数名称和常量可以不按 CREATE PROCEDURE 语句中定义的顺序出现。

（5）value——过程中参数的值。如果参数名称没有指定，则参数值必须以 CREATE PROCEDURE 语句中定义的顺序给出。

（6）@variable——用来保存参数或者返回参数的变量。

（7）OUTPUT——指定存储过程必须返回一个参数。该存储过程的匹配参数也必须由关键字 OUTPUT 创建。

（8）WITH RECOMPILE——强制编译新的计划。如果所提供的参数为非典型参数或者数据有很大的改变，则使用该选项，在以后的程序执行中使用更改过的计划。

【例 7-6】 执行存储过程 student_avg。

```
EXECUTE student_avg
```

执行结果如图 7-7 所示。

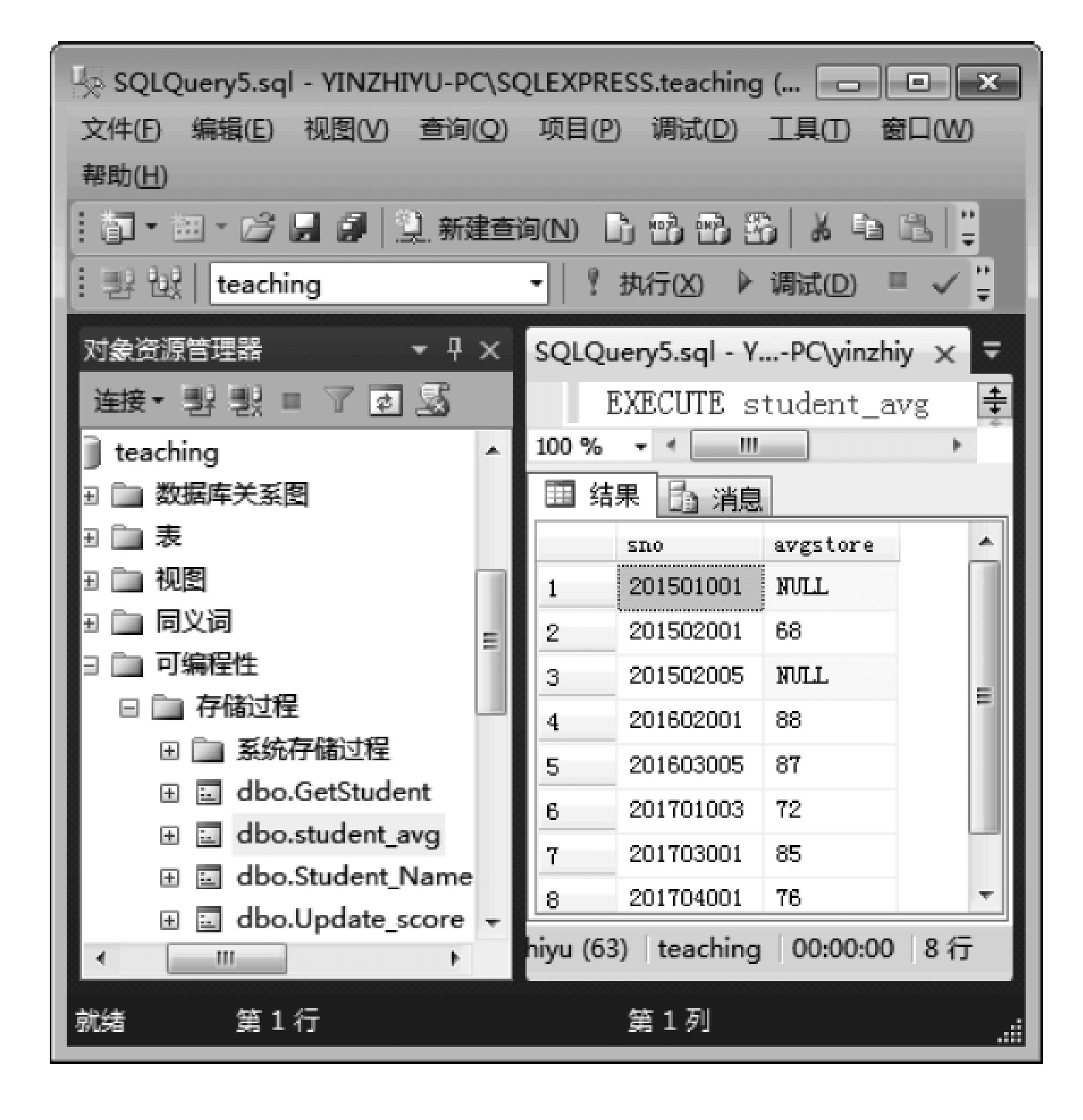


图 7-7　执行存储过程 student_avg

【例 7-7】 执行带参数的存储过程 GetStudent，查询 201501001 号学生的基本信息。

```
EXECUTE GetStudent '201501001'
```

【例 7-8】 执行修改成绩的存储过程 Update_score，将 201602001 号学生选修的 C001 号课程的成绩改为 100。

```
EXECUTE Update_score '201602001','C001',100
```

【例 7-9】 用账号“41254280033512010000”执行存储过程 add_10。

```
EXECUTE add_10 '41254280033512010000'
```

【例 7-10】 执行带有输入和输出参数的存储过程 s_balance。

```
Declare @IDNO char(18),@dname nvarchar(10),@sbalance money
EXECUTE s_balance '133 * * * 198708150101',@dname OUTPUT,@sbalanceOUTPUT
Print '储户' + @dname + '目前总余额' + str(@sbalance)
```

执行结果如图 7-8 所示。

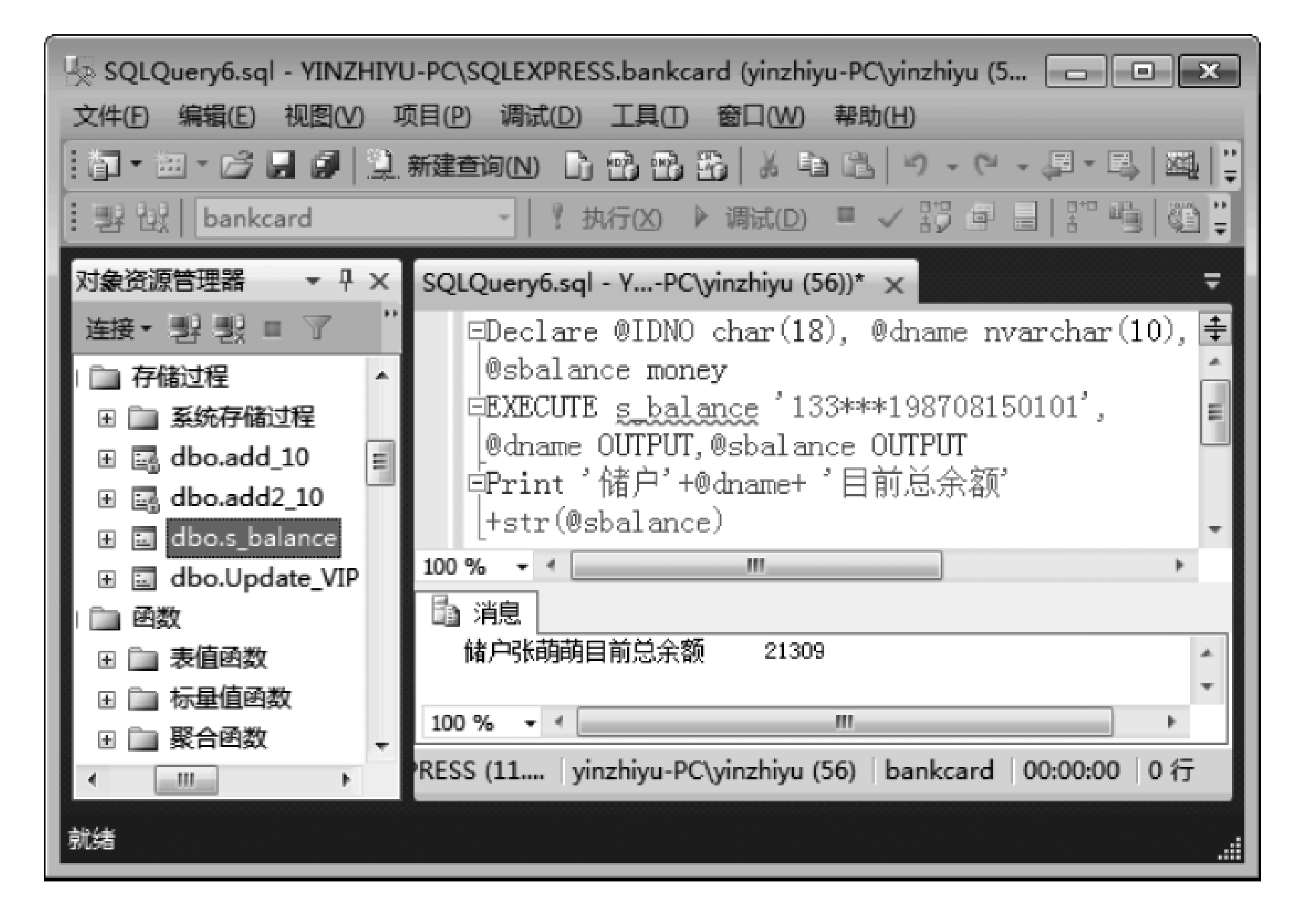


图 7-8　存储过程 s_balance 的执行结果

7.1.5　查看存储过程

查看存储过程可以使用 SSMS 界面,也可以使用 T-SQL 语句。

1. 使用 SSMS 查看存储过程

(1) 打开 SSMS,展开存储过程所在的数据库,展开"可编程性"选项,右击存储过程名,如 teaching 中的 GetStudent,在弹出的快捷菜单中选择"编写存储过程脚本为"→"CREATE 到"→"新查询编辑器窗口"命令,如图 7-9 所示。

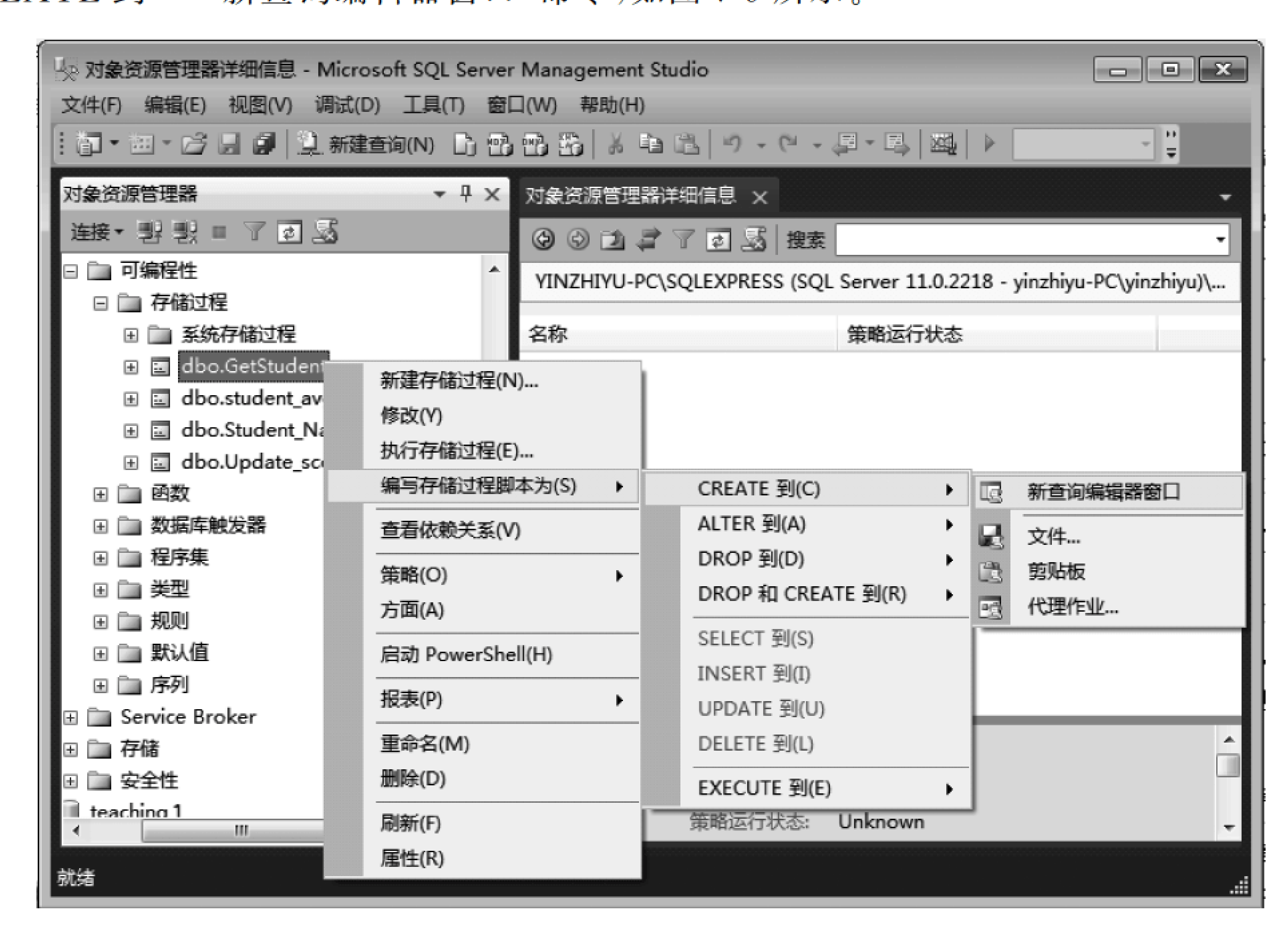


图 7-9　查看存储过程的菜单命令

(2) 进入"查询编辑器"窗口,可以看到 CREATE PROCEDURE 代码,如图 7-10 所示。

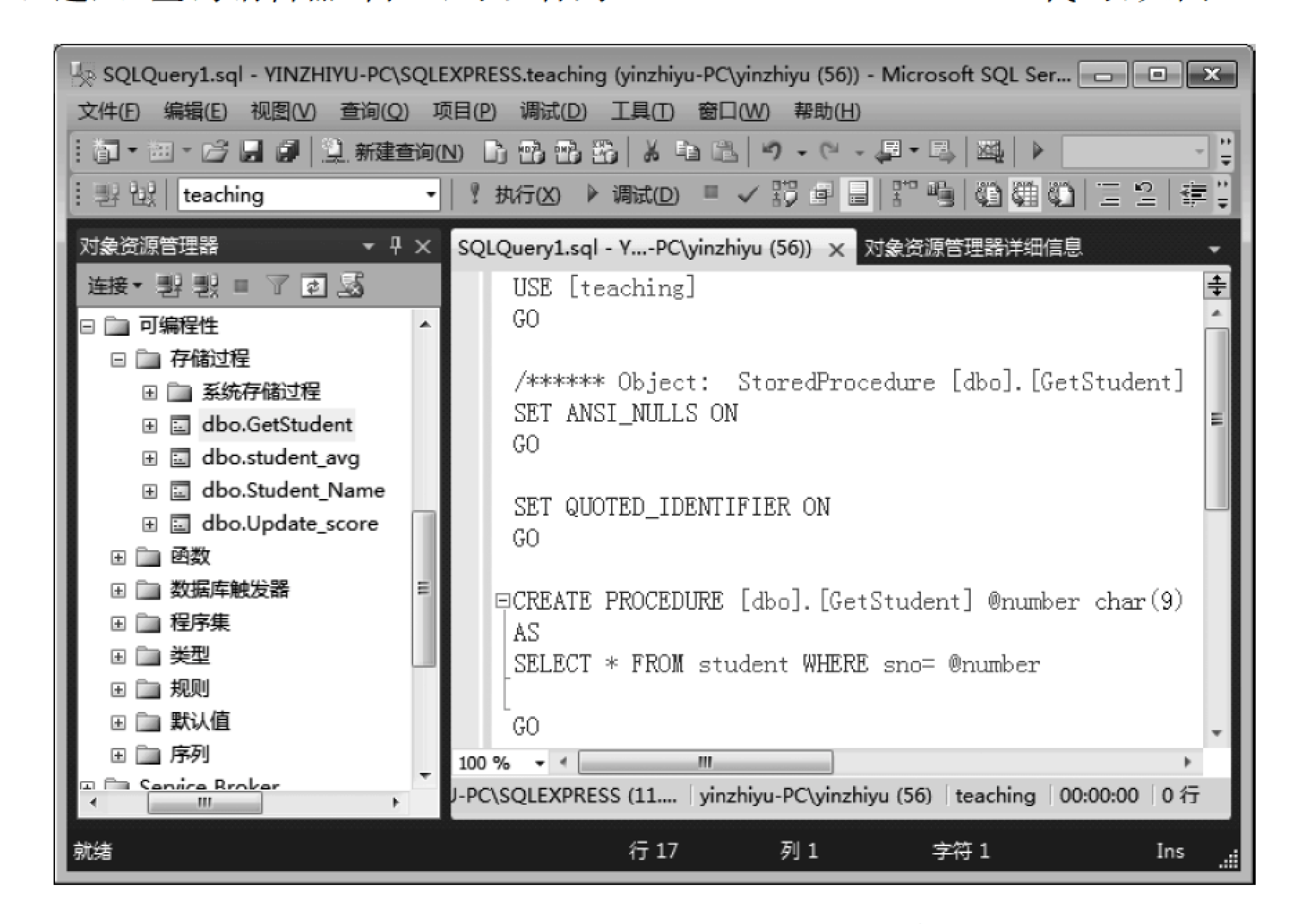


图 7-10　查看 CREATE PROCEDURE 代码

2. 使用 T-SQL 语句查看存储过程

可以执行系统存储过程 sp_helptext,以查看创建存储过程的命令语句;也可以执行系统存储过程 sp_help,以查看存储过程的名称、拥有者、类型、创建时间,以及存储过程中所使用的参数信息。其语法格式分别为:

```
sp_helptext 存储过程名称
sp_help 存储过程名称
```

【例 7-11】　查看存储过程 s_balance 的相关信息。

(1) sp_helptext s_balance

执行结果如图 7-11 所示。



图 7-11　用 sp_helptext 查看存储过程 s_balance

(2) sp_help s_balance

执行结果如图 7-12 所示。

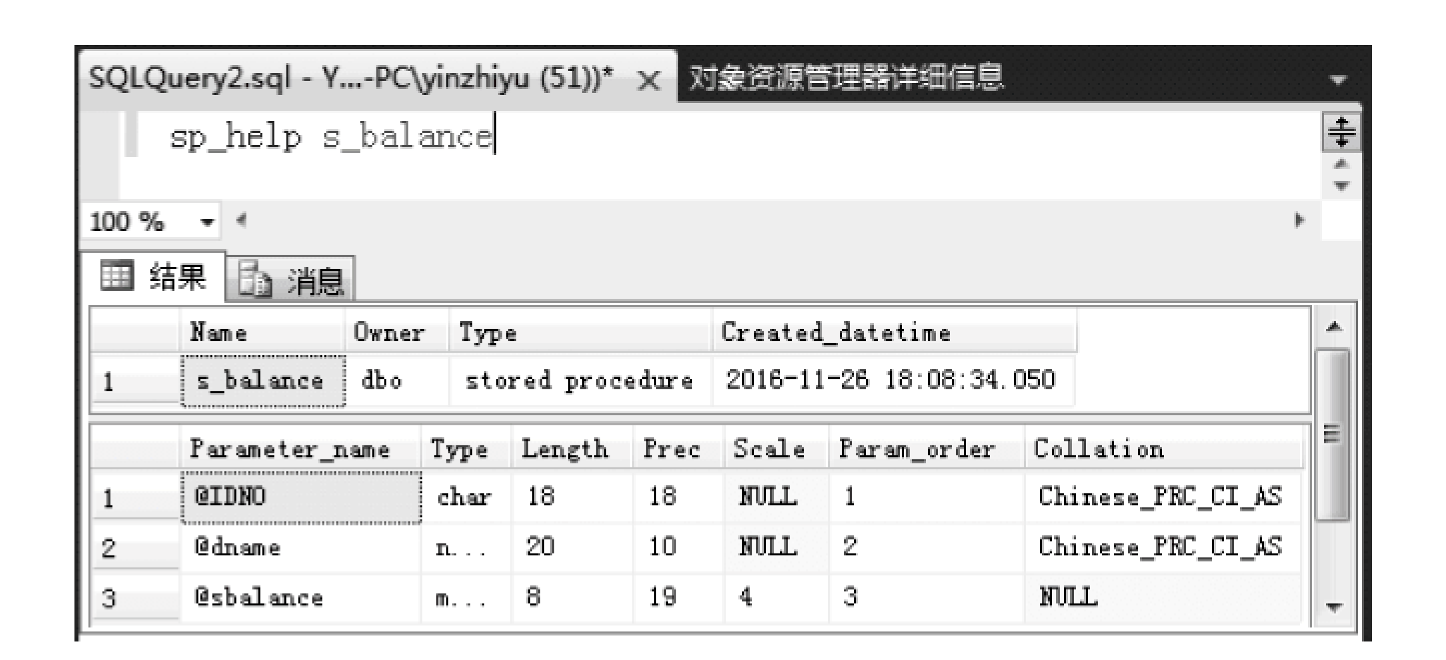


图 7-12　用 sp_help 查看存储过程 s_balance

7.1.6　修改和删除存储过程

1. 修改存储过程

修改存储过程可以在 SSMS 中右击要修改的存储过程，选择“修改”命令进行，具体与创建时的步骤基本相同；或者，也可以通过 T-SQL 中的 ALTER 语句来完成。

ALTER 语句的语法格式如下：

```
ALTER { PROC | PROCEDURE } procedure_name [ ; number ]
   [ { @parameter data_type }
   [ VARYING ] [ = default ] [ [ OUT [ PUT ] ] [ ,…n ]
 [ WITH { RECOMPILE | ENCRYPTION  | RECOMPILE , ENCRYPTION } [ ,…n ] ]
 [FOR REPLICATION]
 AS   sql_statement [ …n ]
```

注：语句中的参数与 CREATE PROCEDURE 语句中的参数相同。

【例 7-12】 修改存储过程 add_10，将 3000 元和 10 元设置为两个参数的默认值，使存储过程应用更灵活。

```
USE bankcard
GO
ALTER PROCEDURE add_10@AccNO char(20),@exp money = 3000,@add int = 10
WITH ENCRYPTION
AS
IF (SELECT SUM(Expense) FROM Trecord
WHERE TDate = CONVERT(varchar(10),GETDATE(), 120)
AND AccNO = @AccNO)> = @exp
  BEGIN
   UPDATE Account SET Balance = Balance + @add WHERE AccNO = @AccNO
   INSERT Trecord(TDate,AccNO,Income,Abstract) VALUES
   (CONVERT(varchar(10),GETDATE(), 120),@AccNO,@add, '银行活动奖励')
  END
```

【例 7-13】 执行带有参数和默认值的存储过程 add_10。

(1) EXECUTE add_10 '4125428003351201 0000'

(2) EXECUTE add_10 '41254280033512010000',2000

(3) EXECUTE add_10 '41254280033512010000',5000,20

2. 删除存储过程

对于不需要的存储过程，可以在 SSMS 中右击要删除的存储过程，选择“删除”命令将其删除；也可以使用 T-SQL 语句中的 DROP PROCEDURE 命令将其删除。如果另一个存储过程调用某个已删除的存储过程，则 SQL Server 2008 会在执行该调用过程时显示一条错误信息。如果定义了同名和参数相同的新存储过程来替换已删除存储过程，那么引用该过程的其他过程仍能顺利执行。

删除存储过程的 T-SQL 语句的语法格式为：

```
DROP PROCEDURE {procedure_name} [,...n]
```

其中，procedure_name 指要删除的存储过程或存储过程组的名称。

【例 7-14】 删除存储过程 s_balance。

```
DROP PROCEDURE s_balance
```

7.2 触 发 器

就本质而言，触发器也是一种存储过程，它在特定语言事件发生时自动执行。

7.2.1 触发器概述

在 SQL Server 2008 数据库系统中，存储过程和触发器都是 SQL 语句和流程控制语句的集合。就本质而言，触发器也是一种存储过程，它是一种在基本表被修改时自动执行的内嵌过程，主要通过事件进行触发而被执行，而存储过程可以通过存储过程的名字而被直接调用。

当对某一张表进行诸如 UPDATE、INSERT、DELETE 等操作时，SQL Server 2008 就会自动执行触发器所定义的 SQL 语句，从而确保对数据的处理符合由这些 SQL 语句所定义的规则。触发器的主要作用是实现主码和外键所不能保证的复杂的参照完整性和数据的一致性，从而有助于强制引用完整性，以便在添加、更新或删除表中的行时保留表之间已定义的关系。

由于在触发器中可以包含复杂的处理逻辑，因此，应该将触发器用来保持低级的数据的完整性，而不是返回大量的查询结果。

使用触发器主要可以实现以下操作：

(1) 强制实现比 CHECK 约束更复杂的数据完整性。

在数据库中要实现数据的完整性的约束，可以使用 CHECK 约束或触发器来实现。但是在 CHECK 约束中不允许引用其他表中的列来完成检查工作，而触发器可以引用其他表中的列来完成数据的完整性的约束。

(2) 使用自定义的错误提示信息。

用户有时需要在数据的完整性遭到破坏或其他情况下，使用预先自定义好的错误提示信息或动态自定义的错误提示信息。通过使用触发器，用户可以捕获破坏数据的完整性的操作，并返回自定义的错误提示信息。

(3) 实现数据库中多张表的级联修改。

用户可以通过触发器对数据库中的相关表进行级联修改。

(4) 比较数据库修改前后数据的状态。

触发器提供了访问由 INSERT、UPDATE 或 DELETE 语句引起的数据前后状态变化的能力。因此用户可以在触发器中引用由于修改所影响的记录行,并可以阻止数据库中未经许可的更新和变化。

(5) 调用存储过程。

约束的本身是不能调用存储过程的,但是触发器本身就是一种存储过程,而存储过程是可以嵌套使用的,所以触发器也可以调用一个或多个存储过程。

(6) 维护非规范化数据。

用户可以使用触发器来保证非规范数据库中的低级数据的完整性。维护非规范化数据与表的级联是不同的,表的级联指的是不同表之间的主外键关系,维护表的级联可以通过设置表的主码与外键的关系来实现。而非规范数据通常是指在表中派生的、冗余的数据值,维护非规范化数据应该通过使用触发器来实现。

7.2.2 触发器的分类

1. DML 触发器

DML 触发器是当数据库服务器中发生数据操作语言(DML)事件时会自动执行的存储过程。

DML 事件包括在指定表或视图中修改数据的 INSERT 语句、UPDATE 语句或 DELETE 语句。DML 触发器可以查询其他表,还可以包含复杂的 T-SQL 语句。系统将触发器和触发它的语句作为可在触发器内回滚的单个事务对待,如果检测到错误(例如,磁盘空间不足),则整个事务自动回滚。

DML 触发器经常用于强制执行业务规则和数据完整性,也可用于强制引用(参照)完整性,以便在多个表中添加、更新或删除行时,保留在这些表之间所定义的关系。但 SQL Server 通常通过 ALTER TABLE 和 CREATE TABLE 语句来提供声明性引用完整性。引用完整性是指有关表的主码和外键之间的关系的规则。若要强制实现引用完整性,请在 ALTER TABLE 和 CREATE TABLE 中使用 PRIMARY KEY 和 FOREIGN KEY 约束。如果触发器所在的表上存在约束,则在 INSTEAD OF 触发器执行之后、AFTER 触发器执行之前检查这些约束。如果违反了约束,则将回滚 INSTEAD OF 触发器操作,并且不激活 AFTER 触发器。

SQL Server 2008 的 DML 触发器分为两类。

(1) AFTER 触发器。这类触发器在记录已经改变完之后,才会被激活执行,它主要用于记录变更后的处理或检查,一旦发现错误,可以用 ROLLBACK 语句来回滚本次的操作。

以删除记录为例。当 SQL Server 接收到一个要执行删除操作的 SQL 语句时,SQL Server 先将要删除的记录存放在一个临时表(删除表 deleted)里,然后把数据表里的记录删除,再激活 AFTER 触发器,执行 AFTER 触发器里的 SQL 语句。执行完毕之后,删除内存中的删除表 deleted,退出整个操作。

(2) INSTEAD OF 触发器。与 AFTER 触发器不同,这类触发器一般是用来取代原本的操作,在记录变更之前发生,它并不去执行原来 SQL 语句里的操作(UPDATE、INSERT、DELETE),而是执行触发器本身所定义的操作。

2. DDL 触发器

DDL 触发器是 SQL Server 2005 及以后版本新增的一个触发器类型，是一种特殊的触发器，它在响应数据定义语言(DDL)语句时触发，一般用于在数据库中执行管理任务。

添加、删除或修改数据库的对象，一旦误操作，可能会导致大麻烦，需要数据库管理员或开发人员对相关可能受影响的实体进行代码的重写。为了在数据库结构发生变动而出现问题时，能够跟踪问题和定位问题的根源，可以利用 DDL 触发器来记录类似"用户建立表"这种会导致变动的操作，以大大减轻跟踪和定位数据库模式变化的烦琐程度。

与 DML 触发器一样，DDL 触发器也是通过事件激活并执行其中的 SQL 语句的。

但与 DML 触发器不同，DML 触发器是响应 UPDATE、INSERT 或 DELETE 语句而激活的，DDL 触发器则是响应 CREATE、ALTER、DROP、GRANT、DENY、REVOKE 和 UPDATE STATISTICS 等语句而激活的。

一般来说，在以下几种情况下可以使用 DDL 触发器：

(1) 数据库里的库架构或数据表架构很重要，不允许被修改。

(2) 防止数据库或数据表被误操作删除。

(3) 在修改某个数据表结构的同时修改另一个数据表的相应的结构。

(4) 要记录对数据库结构操作的事件。

7.2.3 创建触发器

在创建触发器前，需要注意以下问题：

(1) CREATE TRIGGER 语句必须是批处理中的第一条语句，只能用于一个表或视图。

(2) 创建触发器的权限默认为表的所有者，不能将该权限转给其他用户。

(3) 虽然触发器可以引用当前数据库以外的对象，但只能在当前数据库中创建。

(4) 虽然不能在临时表或系统表上创建触发器，但是触发器可以引用临时表。不应引用系统表，而应使用信息架构视图。

(5) 在含有用 DELETE 或 UPDATE 操作定义的外键的表中，不能定义 INSTEAD OF 触发器。

(6) 虽然 TRUNCATE TABLE 语句类似于没有 WHERE 子句的 DELETE 语句，但不会激发 DELETE 触发器，因为 TRUNCATE TABLE 语句没有记录日志。

创建触发器时需指定以下几项内容：

(1) 触发器的名称。

(2) 在其上定义触发器的表。

(3) 触发器将何时激发。

(4) 激活触发器的数据修改语句，有效选项为 INSERT、UPDATE 或 DELETE，多个数据修改语句可激活同一个触发器。

在 SQL Server 中创建 DML 触发器主要有两种方式：在 SSMS 界面中创建，或通过在查询窗口中执行 T-SQL 语句创建。

1. 在 SSMS 中创建 DML 触发器

在 SSMS 中创建 DML 触发器的步骤如下。

（1）打开 SSMS，展开要创建 DML 触发器的数据库和其中的表或视图（如 student 表），右击“触发器”选项，选择“新建触发器”命令，如图 7-13 所示。

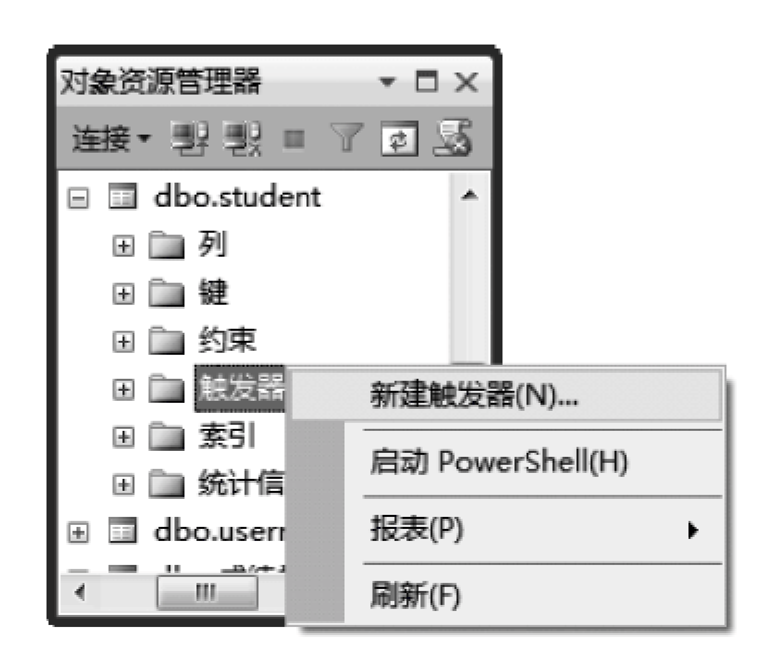


图 7-13　在 SSMS 中创建 DML 触发器

（2）出现创建触发器的 T-SQL 语句，编辑相关的命令即可，如图 7-14 所示。

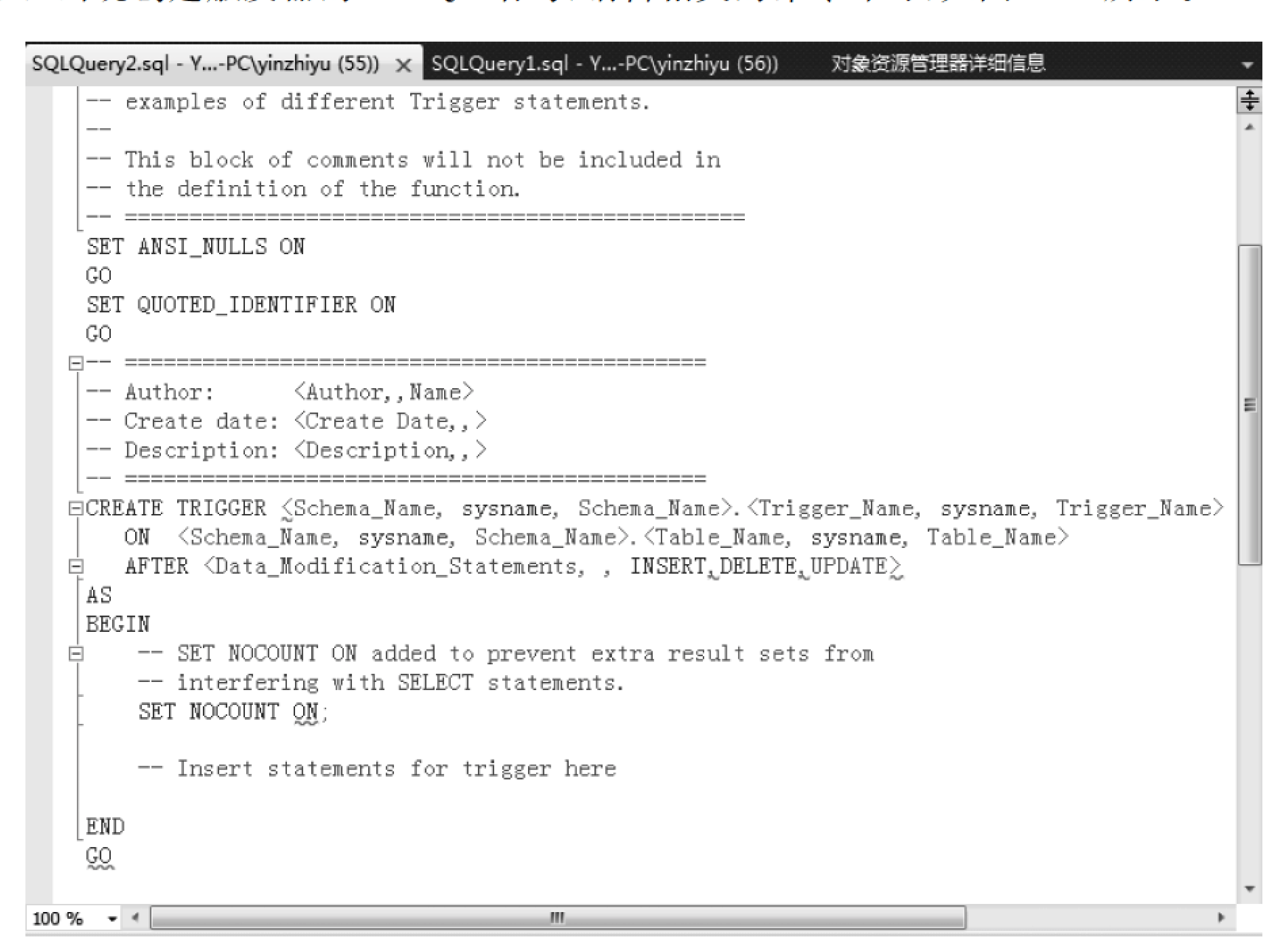


图 7-14　创建 DML 触发器的 T-SQL 语句

（3）命令编辑成功后，进行语法检查，然后单击“！执行”按钮，至此，一个 DML 触发器建立成功。

2. 利用 T-SQL 语句创建触发器

SQL Server 提供了 CREATE TRIGGER 语句来创建触发器。

语法格式如下：

```
CREATE TRIGGER trigger_name
 ON { table_name | view }
 [WITH ENCRYPTION]
 { FOR | AFTER | INSTEAD OF }
```

```
{ [ INSERT ] [ DELETE ] [ UPDATE ] }
[NOT FOR REPLICATION]
AS sql_statement [...n ]
```

其中各参数说明如下：

(1) trigger_name——触发器的名称，触发器名称必须符合标识符规则，并且在数据库中必须唯一。用户可以选择是否指定触发器所有者名称。

(2) table_name | view——要在其上执行触发器的表或视图，可以选择是否指定表或视图的所有者名称。

(3) WITH ENCRYPTION——加密 syscomments 表中包含 CREATE TRIGGER 语句文本的条目。使用 WITH ENCRYPTION 可防止将触发器作为 SQL Server 复制的一部分发布，这是为了满足数据安全的需要。

(4) AFTER——指定触发器只有在触发 SQL 语句中指定的所有操作都已成功执行后才激发。所有的引用级联操作和约束检查也必须成功完成后，才能执行此触发器。如果仅指定 FOR 关键字，则 AFTER 是默认设置。不能在视图上定义 AFTER 触发器。

(5) INSTEAD OF——指定执行触发器而不是执行触发语句，从而替代触发语句的操作。在表或视图上，每个 INSERT、UPDATE 或 DELETE 语句最多可以定义一个 INSTEAD OF 触发器。如果在对一个可更新的视图进行定义时，使用了 WITH CHECK OPTION 选项，则不允许在这个视图上定义 INSTEAD OF 触发器。用户必须用 ALTER VIEW 删除选项后，才能定义 INSTEAD OF 触发器。

对于 INSTEAD OF 触发器，不允许在具有 ON DELETE 级联操作引用关系的表上使用 DELETE 选项。同样，也不允许在具有 ON UPDATE 级联操作引用关系的表上使用 UPDATE 选项。

(6) INSERT、UPDATE、DELETE——指定在表或视图上执行哪些数据修改语句时会激活触发器的关键字。其中必须至少指定一个选项，允许使用以任意顺序组合的关键字，多个选项需要用逗号分隔。

(7) NOT FOR REPLICATION——表示当复制进程更改触发器所涉及的表时，不应执行该触发器。

(8) sql_statement——定义触发器被触发后将执行的数据库操作，它指定触发器执行的条件和动作。触发器条件是除引起触发器执行的操作外的附加条件；触发器动作是指当前用户执行激发触发器的某种操作并满足触发器的附加条件时触发器所执行的动作。

首先举一个在数据库上创建 DDL 触发器的例子。

【例 7-15】 使用 DDL 触发器 limited 来防止数据库中的任一表被修改或删除。

```
USE teaching
GO
CREATE TRIGGER limited ON database
FOR DROP_TABLE, ALTER_TABLE
AS
  PRINT '名为 limited 的触发器不允许您执行对表的修改或删除操作!'
  ROLLBACK
```

以上的 T-SQL 语句执行成功后，在 teaching 中就创建了一个 DDL 触发器 limited。

打开 SSMS,展开 teaching 数据库下的“可编程性”选项,再展开“数据库触发器”就可以看到刚刚创建的触发器 limited,如图 7-15 所示。

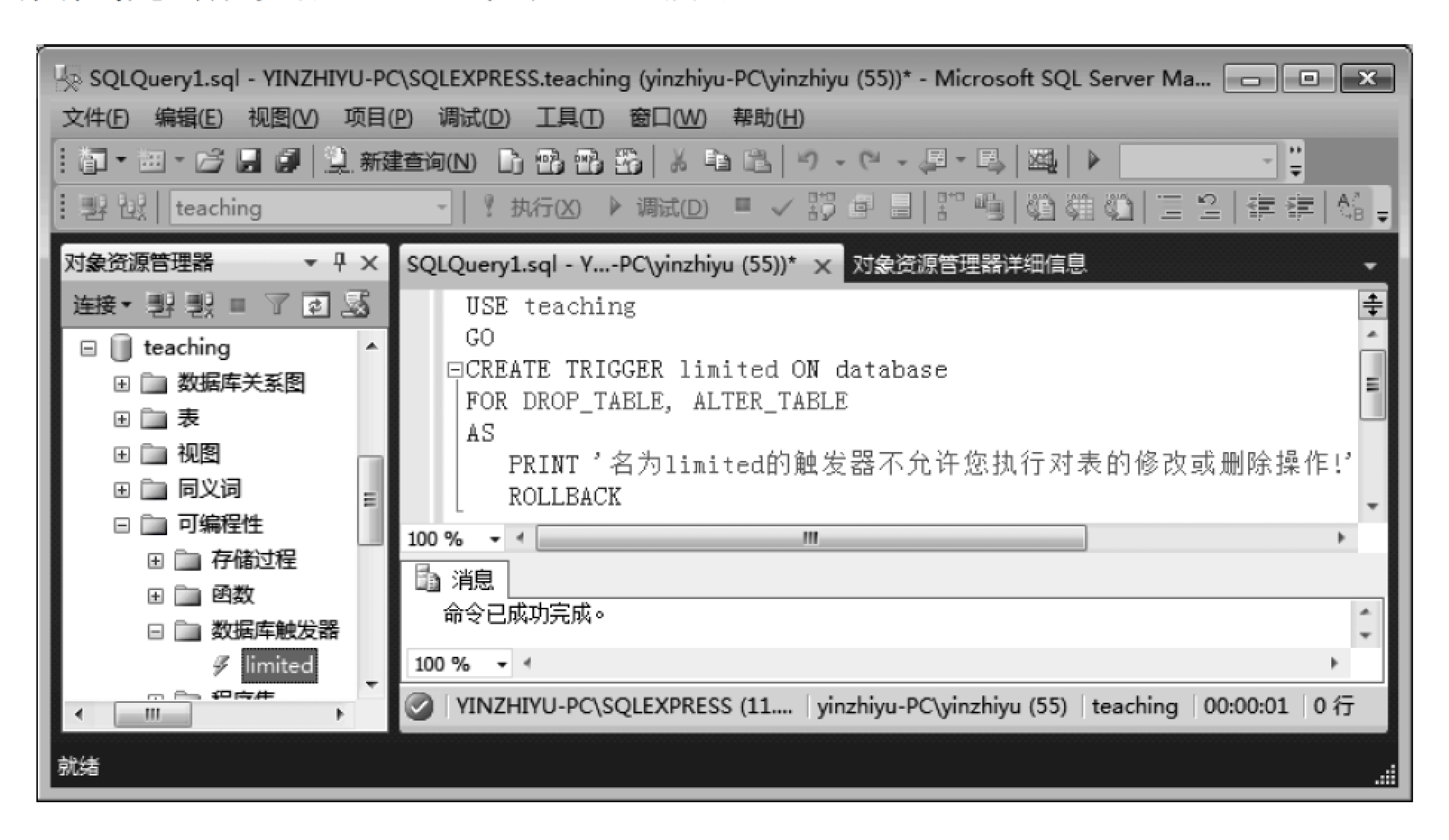


图 7-15 创建好的 DDL 触发器 limited

【例 7-16】 假定有修改 student 表权限的某个用户要修改 student 表,添加一个“年龄”列:age tinyint。

```
ALTER TABLE student
ADD age tinyint
```

执行结果如图 7-16 所示。

事实上,当任一用户在 teaching 数据库中试图修改表的结构或删除表时,都会触发 limited 触发器。该触发器显示提示信息,并回滚用户试图执行的操作。

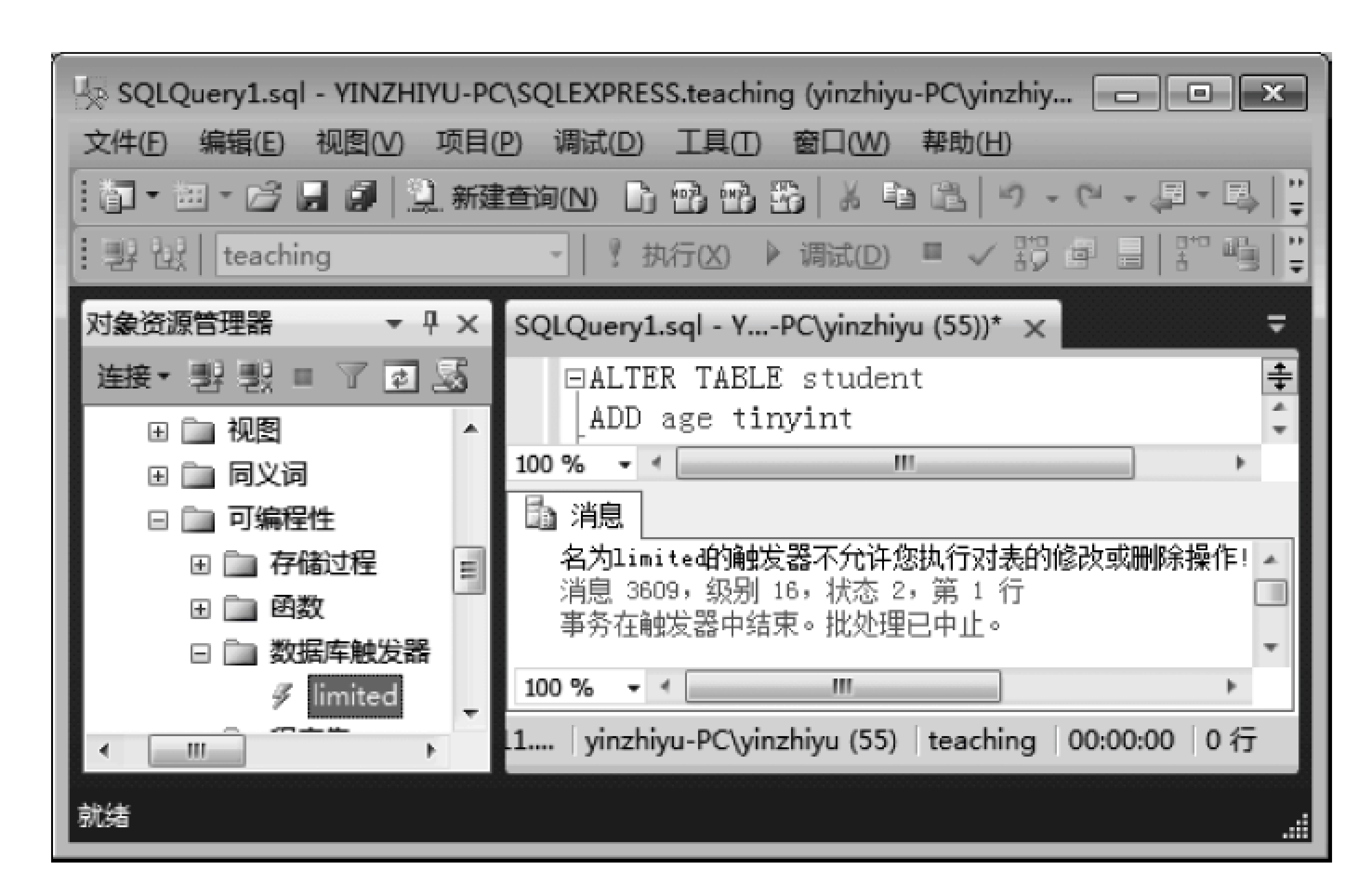


图 7-16 修改 student 表时触发了 limited 触发器

【例 7-17】 为 student 表创建一个简单 DML 触发器,使得在插入和修改数据时,都会自动显示提示信息。

```
USE teaching
GO
CREATE TRIGGER reminder ON student
FOR INSERT,UPDATE
AS print '你在插入或修改 student 表的数据'
```

【例 7-18】 将姓名为“刘梅”的学生的名字改为“刘小梅”。

```
UPDATE student SET sname = '刘小梅' WHERE sname = '刘梅'
```

执行结果如图 7-17 所示。

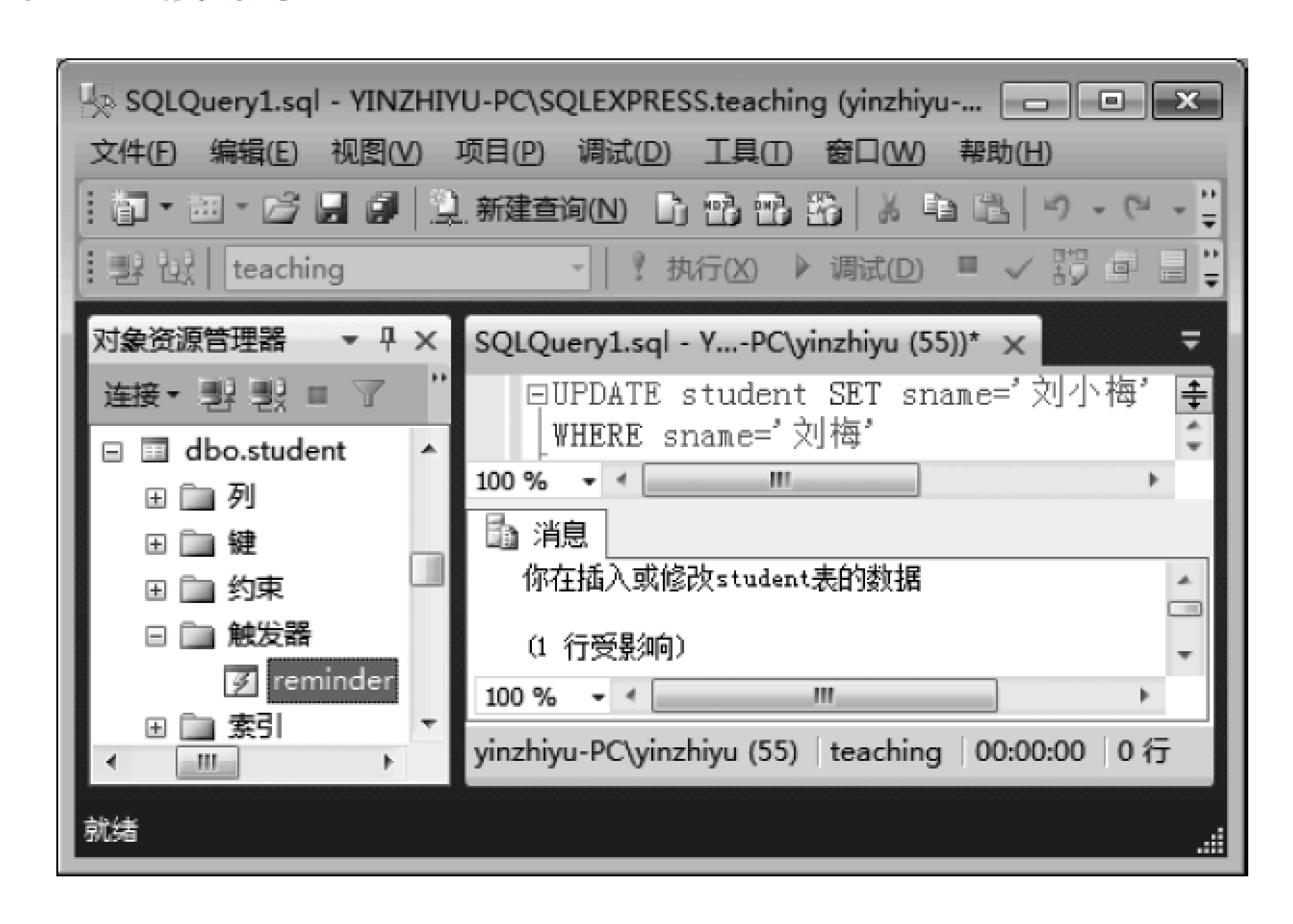


图 7-17 修改学生姓名触发了 reminder 触发器

【例 7-19】 为 student 表创建一个 DML 触发器,使得在插入和修改数据时,都会自动显示所有学生的信息。

```
CREATE TRIGGER print_table ON student
FOR INSERT,UPDATE
AS SELECT * FROM student
```

【例 7-20】 将姓名为“刘小梅”的名字改为“刘梅”。

```
UPDATE student SET sname = '刘梅' WHERE sname = '刘小梅'
```

执行结果如图 7-18 所示。

【例 7-21】 在 student 表上创建一个 DELETE 类型的触发器,使得在删除数据时,显示删除学生的个数。

```
CREATE TRIGGER del_count ON student
FOR DELETE
AS
  DECLARE @count varchar(50)
  SET @count = STR(@@ROWCOUNT) + '个学生被删除'
  SELECT @count
RETURN
```

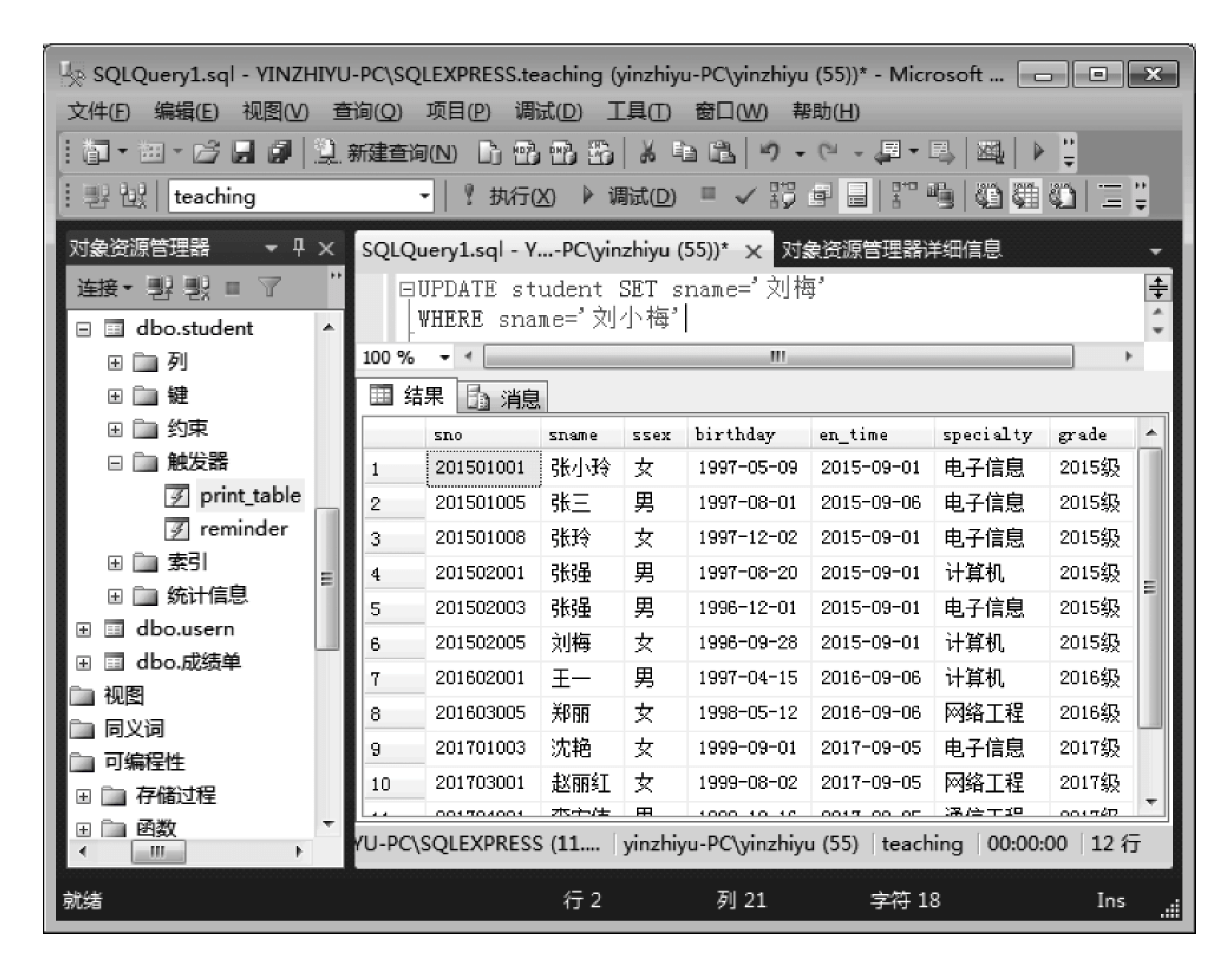


图 7-18　修改学生姓名触发了 print_table 触发器

【例 7-22】 删除所有"计算机"专业的学生，触发 del_count 触发器。

```
DELETE FROM student WHERE specialty = '计算机'
```

在 SQL Server 2008 中，为每个 DML 触发器都定义了两个特殊的表：一个是插入表(Inserted)，一个是删除表(Deleted)。这两个表是建立在数据库服务器的内存中的，是由系统管理的逻辑表，而不是真正存储在数据库中的物理表。对于这两个表，用户只有读取的权限，没有修改的权限。

在触发器的执行过程中，SQL Server 建立和管理这两个临时表。这两个表的结构与触发器所在数据表的结构是完全一致的，其中包含了在激发触发器的操作中插入或删除的所有记录。当触发器的工作完成之后，这两个表也将会从内存中删除。

插入表中存放的是更新后的记录：对于插入记录操作来说，插入表里存放的是要插入的数据；对于更新记录操作来说，插入表中存放的是更新后的记录。

删除表中存放的是更新前的记录：对于更新记录操作来说，删除表中存放的是更新前的记录(更新完后即被删除)；对于删除记录操作来说，删除表中存入的是被删除的旧记录。

也就是说，在用户执行 INSERT 语句时，所有被添加的记录都会存储在插入表和触发程序表中；在用户执行 DELETE 语句时，从触发程序表中被删除的行会发送到删除表；对于 UPDATE 语句，SQL Server 先将要进行修改的记录存储到删除表中，然后再将修改后的数据插入到插入表以及触发程序表。

下面利用触发器和这两个特殊的表实现级联式数据修改。

【例 7-23】 在 bankcard 数据库中使用流程控制语句创建 Transactions 触发器。当向交易记录表(Trecord)中添加一条交易信息时，如果今天某账号交易支出了某个金额，则其账户余额减去此金额；如果今天某账号收入了某个金额，则其账户余额加上此金额。

```
USE bankcard
GO
CREATE TRIGGER Transactions ON Trecord
FOR INSERT
AS
IF (SELECT Expense FROM inserted) IS NOT NULL
UPDATE account SET Balance = Balance - (SELECT Expense FROM inserted)
WHERE AccNO = (SELECT AccNO FROM inserted)
ELSE
UPDATE account SET Balance = Balance + (SELECT Income FROM inserted)
WHERE AccNO = (SELECT AccNO FROM inserted)
```

【例 7-24】 向交易记录表(Trecord)中添加一条交易信息记录，触发 Transactions 触发器。

```
INSERT Trecord VALUES(getdate(),'41254280033512060000',50,NULL,'43674280033512050000','北国
超市','消费支出')
```

如果表和表之间存在主、外键约束，那么可以通过 CREATE TABLE 或 ALTER TABLE 命令或者通过设置关系图的属性设置级联修改和级联删除；当然也可以通过触发器编程来实现级联修改和级联删除。

触发器可以实现复杂的约束和特殊的约束。

下面用一个详细的例子来介绍。

首先在 teaching 数据库中创建 3 个表。

(1) 教师表：包括教师号、姓名和职称属性。

```
CREATE TABLE teacher
( tno int primary key,
 sname char(6),
 prof_title char(10) )
GO
```

(2) 教师工资表：包括教师号、姓名和工资属性。

```
CREATE TABLE teacher_salary
(tno int primary key foreign key references teacher(tno),
 sname char(6),
 salary int  )
GO
```

(3) 工资级别表：包括职称、最低工资和最高工资属性。

```
CREATE TABLE salary_level
(prof_title char(10) primary key ,
 minsalary int,
 maxsalary int  )
```

```
GO
```

之后插入如下数据：

```
INSERT teacher VALUES(1,'郑浩','教授')
INSERT teacher VALUES(2,'王伟','副教授')
INSERT teacher VALUES(3,'李平','讲师')
INSERT salary_level VALUES('教授',8000,9900)
INSERT salary_level VALUES('副教授',5900,7700)
INSERT salary_level VALUES('讲师',4500,5900)
INSERT salary_level VALUES('助教',3900,4900)
```

【例 7-25】 在教师工资表上创建一个触发器，用于实现复杂的约束：在对教师的工资进行录入和修改时，按职称级别进行约束。

```
CREATE TRIGGER teacher_sala1 ON teacher_salary
FOR INSERT, UPDATE
AS
DECLARE @minsalary int, @maxsalary int, @salary int, @prof varchar(10), @tname varchar(10)
SELECT @minsalary = minsalary, @maxsalary = maxsalary, @salary  = i.salary,
@prof = t.prof_title, @tname = i.sname
FROM inserted i,salary_level s,teacher t
WHERE s.prof_title  =  t.prof_title and t.tno = i.tno
IF NOT (@salary BETWEEN @minsalary AND @maxsalary)
BEGIN
  PRINT @tname + '的职称为:' + @prof + '工资应该在' + str(@minsalary) +
'到' + str(@maxsalary) + '之间.'
   ROLLBACK
END
```

利用命令触发该触发器：

```
INSERT teacher_salary VALUES(1,'郑浩',7800)
```

执行结果如图 7-19 所示。

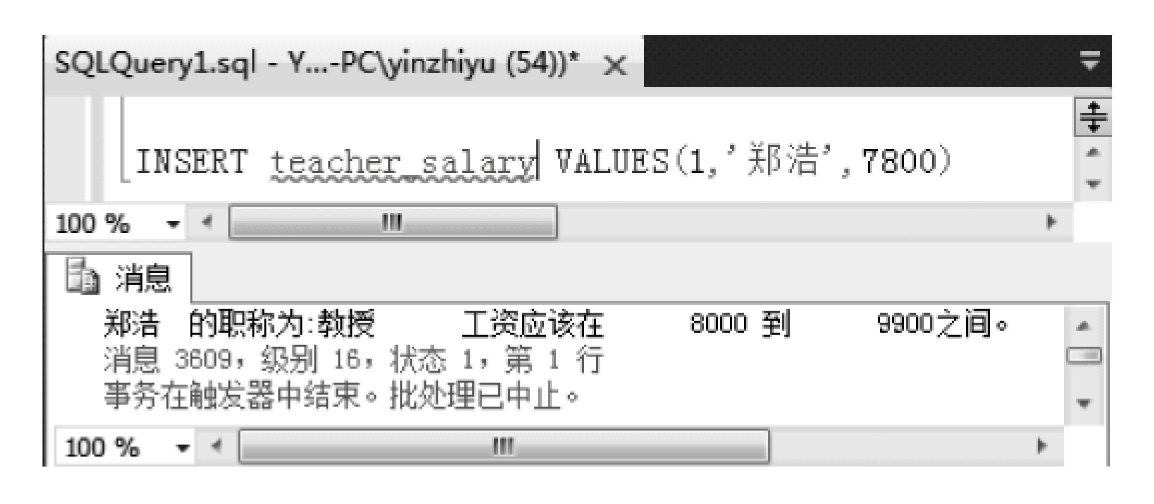


图 7-19 插入违反触发器规则的数据

【例 7-26】 在教师工资表上创建一个触发器，用于实现特殊的约束：规定每月的 10 日前发工资，即对教师的工资进行录入时，触发此触发器，时间不对不能录入。

```
CREATE TRIGGER teacher_sala2 ON teacher_salary
FOR INSERT
AS
declare @d int
```

```
set @d = day(getdate())
IF @d > 10
BEGIN
PRINT '只能在每月的 10 号以前发工资,今天是' + str(@d) + '号.'
ROLLBACK
END
```

利用下面命令触发该触发器：

```
INSERT teacher_salary VALUES(2,'王伟',6200)
```

执行结果如图 7-20 所示。



图 7-20　插入违反触发器规则的数据

【例 7-27】　触发器中调用存储过程。

首先创建一个存储过程 p1 如下：

```
CREATE PROC p1 AS
SELECT * FROM student
```

然后，为 student 表创建一个触发器 tr1；在插入、修改或删除数据时，都会触发此触发器，调用存储过程 p1。

```
CREATE TRIGGER tr1 ON student
FOR INSERT,UPDATE,DELETE
AS EXEC p1
```

【例 7-28】　针对某些列实施监控的列级触发器。

首先，建立登记修改人账号的表 change_user，表结构如图 7-21 所示。

列名	数据类型	允许空
ch_datetime	datetime	☑
ch_column	varchar(20)	☑
ch_name	varchar(20)	☑

图 7-21　change_user 表的表结构

创建触发器 tr_change，用于登记修改数据者及修改时间等信息。

```
CREATE TRIGGER tr_change
ON COURSE FOR UPDATE
AS
IF UPDATE (classhour)
BEGIN
  INSERT change_user
  VALUES (getdate(),'course.classhour',user_name())
```

```
END
ELSE IF UPDATE (credit)
  BEGIN
    INSERT change_user
    VALUES(getdate(),'course.credit',user_name())
  END
```

利用下面命令触发该触发器：

```
UPDATE course SET classhour = 5 WHERE cno = 'C004'
```

执行结果如图 7-22 所示。

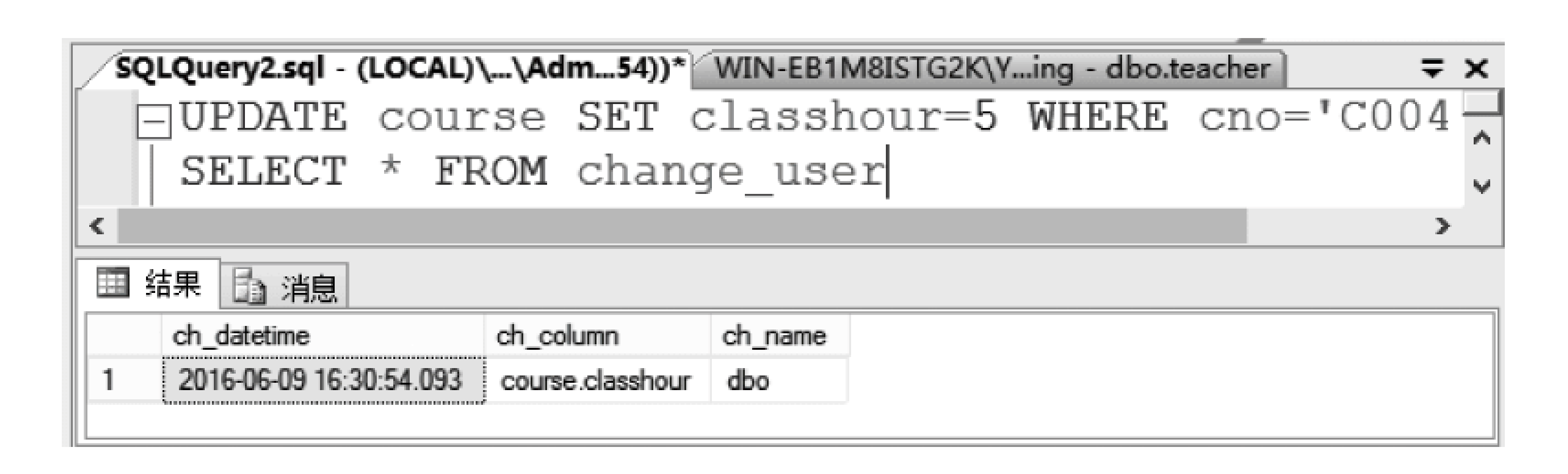


图 7-22　修改 course 表数据后 change_user 的变化

在 teaching 数据库中的其他表上也可以建立相似的触发器，用于登记修改数据者及修改时间等信息，读者可以自己去完成。

7.2.4　查看触发器信息及修改触发器

在 SQL Server 2008 中，一般有两种方法来查看触发器信息：在 SSMS 中查看触发器和使用系统存储过程查看触发器。

1. 在 SSMS 中查看触发器

在 SSMS 中查看触发器信息的具体步骤如下。

(1) 在 SSMS 的"对象资源管理器"面板中，展开 teaching 选项，再展开"表"选项，选中 dbo.student 选项并展开，最后展开"触发器"选项，选中要查看的触发器名，如 del_count，右击，在弹出的快捷菜单中选择"编写触发器脚本为"→"CREATE 到"→"新查询编辑器窗口"命令，如图 7-23 所示。

(2) 在弹出的 T-SQL 命令窗口中显示了该触发器的语句内容，如图 7-24 所示。

2. 使用系统存储过程查看触发器

系统存储过程 SP_HELP 和 SP_HELPTEXT 分别提供了有关触发器的不同信息。

(1) 通过 SP_HELP 系统存储过程，可以了解触发器的一般信息，包括名字、拥有者名称、类型、创建时间。

【例 7-29】 通过 SP_HELP 查看 student 上的触发器 print_table。

```
SP_HELP print_table
```

执行结果如图 7-25 所示。

(2) 通过 SP_HELPTEXT 能够查看触发器的定义信息。

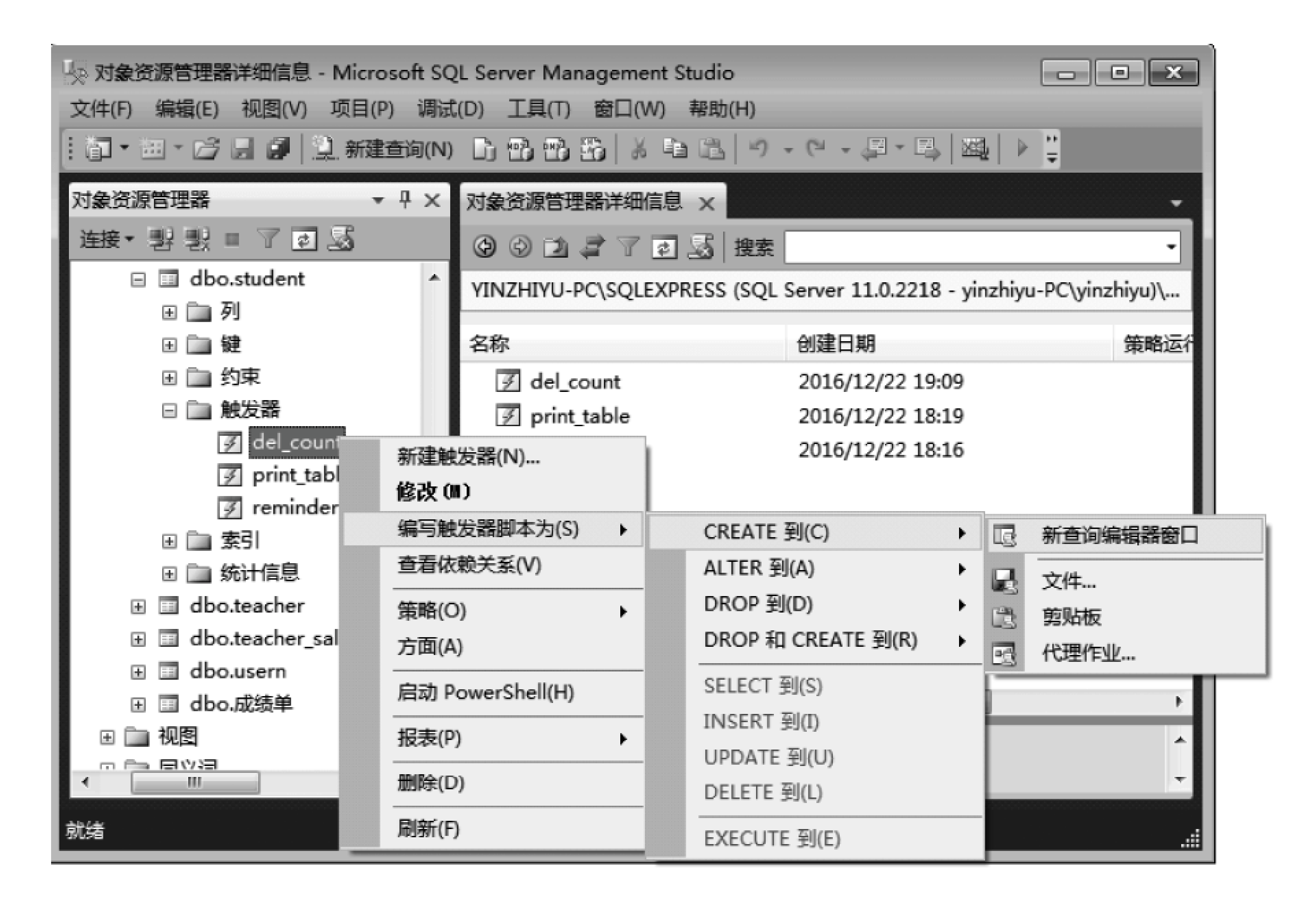


图 7-23　查看触发器菜单

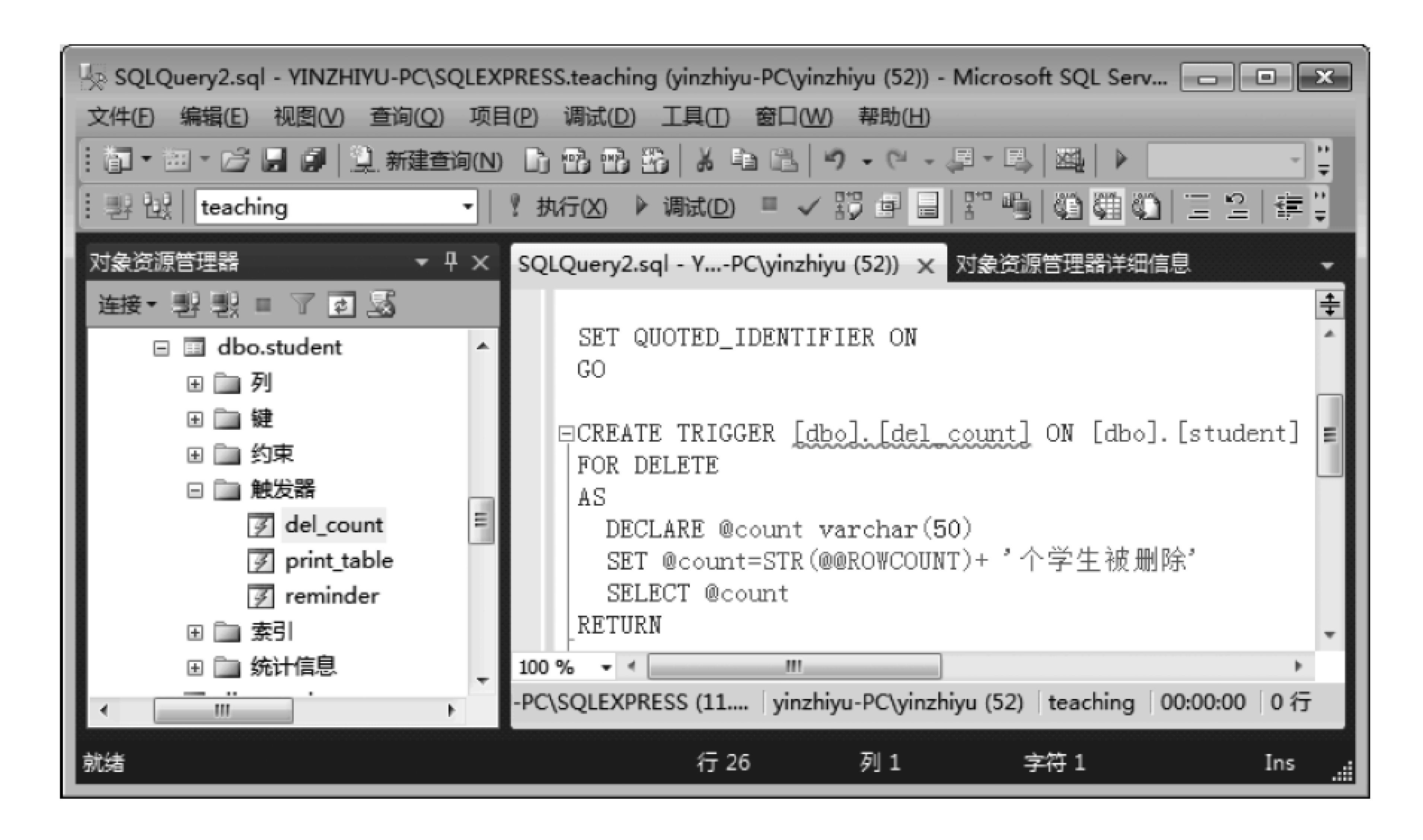
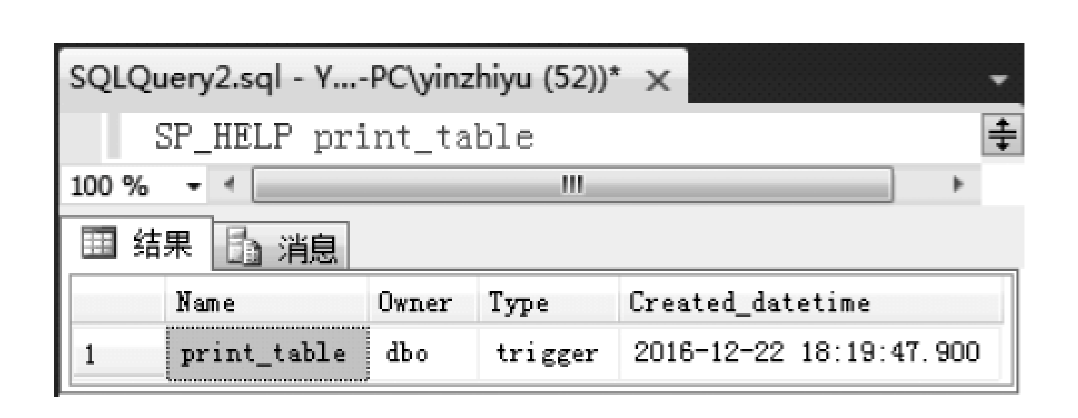


图 7-24　查看 CREATE TRIGGER 创建触发器命令



	Name	Owner	Type	Created_datetime
1	print_table	dbo	trigger	2016-12-22 18:19:47.900

图 7-25　SP_HELP 查看触发器 print_table

【例 7-30】 通过 SP_HELPTEXT 查看 student 上的触发器 print_table。

```
SP_HELPTEXT print_table
```

执行结果如图 7-26 所示。

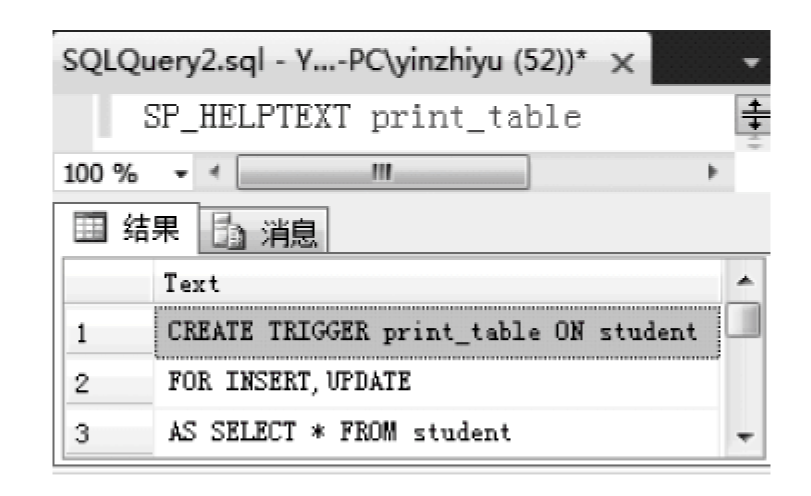


图 7-26 SP_HELPTEXT 查看触发器 print_table

还可以通过使用系统存储过程 SP_HELPTRIGGER 来查看某个特定表上存在的触发器的某些相关信息。

【例 7-31】 通过 SP_HELPTRIGGER 查看 student 表上的触发器信息。

```
SP_HELPTRIGGER student
```

执行结果如图 7-27 所示。

SQLQuery2.sql - Y...-PC\yinzhiyu (52))*

SP_HELPTRIGGER student

100 %

结果 消息

	trigger_name	trigger_owner	isupdate	isdelete	isinsert	isafter	isinsteadof	trigger_schem
1	reminder	dbo	1	0	1	1	0	dbo
2	print_table	dbo	1	0	1	1	0	dbo
3	del_count	dbo	0	1	0	1	0	dbo
4	tr1	dbo	1	1	1	1	0	dbo

图 7-27 SP_HELPTRIGGER 查看 student 表上的触发器

3. 修改触发器

通过使用 SSMS 窗口或 T-SQL 语句,可以修改触发器。

使用 SSMS 窗口修改触发器:在 SSMS 的“对象资源管理器”面板中,展开 teaching 选项,再展开“表”选项,选中 dbo. student 选项并展开,最后再展开“触发器”选项,选中要修改的触发器名,如 del_count,右击,在弹出的快捷菜单中选择“修改”命令,出现创建触发器的 T-SQL 语句,修改相关的命令即可。

通过使用 SQL Server 2008 提供的 ALTER TRIGGER 语句来修改触发器,其语法格式如下:

```
ALTER TRIGGER trigger_name
  ON { table_name | view }
   [WITH ENCRYPTION ]
{ FOR | AFTER | INSTEAD OF }
   { [ INSERT ] [ DELETE ] [ UPDATE ] }
[NOT FOR REPLICATION ]
```

```
AS sql_statement [ ...n ]
```

注：语句中的参数与 CREATE TRIGGER 语句中的参数相同。

【例 7-32】 修改 teaching 数据库中 student 表上的触发器 reminder，使得在用户执行添加或修改操作时，自动给出错误提示信息，撤销此次操作。

```
ALTER TRIGGER reminder
ON student
INSTEAD OF INSERT , UPDATE
AS print '你执行的添加或修改操作无效!'
```

7.2.5 禁止、启用和删除触发器

禁用触发器与删除触发器不同，禁用触发器时，仍会为数据表定义该触发器，只是在执行 INSERT、UPDATE 或 DELETE 语句时，除非重新启用该触发器，否则不会执行该触发器中的操作；而删除触发器是将该触发器在数据表上的定义完全删除，以后如果想使用此触发器，需重新创建。

1. 禁止和启用触发器

在使用触发器时，用户可能会遇到需要禁止某个触发器起作用的场合，例如，在某些表上不允许批量更新操作，所使用的触发器会根据@@ROWCOUNT 来进行判断，如果@@ROWCOUNT 大于预设的值就不允许更新，但是作为数据库管理员难免有批量更新的要求，此时就需要让该触发器不起作用，即禁止。

当一个触发器被禁止时，该触发器仍然存在于表上，只是触发器的动作将不再执行，直到该触发器被重新启用。ALTER TABLE 可以禁止和启用一个表上的一个或者全部的触发器，禁止和启用触发器的语法格式如下：

```
ALTER TABLE table_name
  [ENABLE | DISABLE] TRIGGER
  [ ALL | trigger_name [ ,...n ] ]
```

其中各参数说明如下：

(1) ENABLE | DISABLE TRIGGER——指定启用或禁止触发器。当一个触发器被禁止时，它对表的定义依然存在，但当在表上执行 INSERT、UPDATE 或 DELETE 语句时，该触发器中的操作将不执行，除非重新启用该触发器。

(2) ALL——指定启用或禁止表上所有的触发器。

(3) trigger_name——指定要启用或禁止的一个或几个触发器的名称。

【例 7-33】 禁止 student 表上创建的所有触发器。

```
ALTER TABLE student
DISABLE TRIGGER ALL
```

2. 删除触发器

删除已创建的触发器一般有两种方法。

(1) 在 SSMS 的“对象资源管理器”中找到相应的触发器，右击，在弹出的快捷菜单中选择“删除”命令即可。

(2) 使用 T-SQL 命令 DROP TRIGGER 删除指定的触发器，具体的语法格式如下：

```
DROP TRIGGER trigger_ name
```

【例 7-34】 使用 DROP TRIGGER 命令删除 student 表上的 del_count 触发器。

```
USE teaching
GO
DROP TRIGGER del_count
```

注： 删除触发器所在的表时，SQL Server 将自动删除与该表相关的触发器。

习　题　7

1. 简述存储过程和触发器的优点。
2. 简述存储过程和触发器的分类。
3. 创建一个存储过程，计算指定学生(姓名)的总成绩，存储过程中使用一个输入参数(姓名)和一个输出参数(总成绩)，然后调用该存储过程。
4. 在 bankcard 数据库中创建存储过程 Update_VIP，将账户总余额小于某个值的储户的 VIP 列设置为"否"，大于等于这个值的储户 VIP 列设置为"是"，然后调用该存储过程，例如：

```
EXECUTE Update_VIP 10000
```

5. 在 bankcard 数据库中创建存储过程，查询某账号在某个时间段(某年某月某日至某年某月某日)的交易明细，然后调用该存储过程。
6. 为 student 表创建一个实现级联删除的触发器，当执行删除时，激活该触发器同时删除 sc 表中相应(相同学号的)记录，然后用相关命令语句触发此触发器。
7. 在 teaching 数据库中创建一个学生党费表 st_dues，属性为 sno、sname、dues，含义分别为学号、姓名、党费。sno 是主码，也是外键(参考 student 表的 sno)；创建一个触发器，保证只能在每年的 6 月和 12 月交党费，如果在其他时间录入则显示提示信息，然后用相关命令语句触发此触发器。
8. 修改例 7-23 的触发器，在对 account 表进行余额修改时，如果其卡种类为"借记卡"，则余额不能小于 0，然后用相关命令语句触发此触发器。

第 8 章　事务与并发控制

关系型数据库有四个显著的特征,即安全性、完整性、并发性和监测性。

数据库的安全性就是要保证数据库中数据的安全,防止未授权用户随意修改数据库中的数据,确保数据的安全。在大多数数据库管理系统中,主要是通过许可来保证数据库的安全性。

完整性是数据库的一个重要特征,也是保证数据库中的数据切实有效、防止错误、实现商业规则的一种重要机制。在数据库中,区别所保存的数据是无用的垃圾还是有价值的信息,主要是依据数据库的完整性是否健全。在 SQL Server 中,数据的完整性是通过一系列逻辑来保障的,这些逻辑分为三个方面,即实体完整性、域完整性和参照完整性。

对任何系统都可以这样说,没有监测,就没有优化。只有通过对数据库进行全面的性能监测,才能发现影响系统性能的因素和瓶颈,也才能针对瓶颈因素,采取切合实际的策略,解决问题,提高系统的性能。

为了充分利用数据库资源,发挥数据库共享资源的特点,应该允许多个用户并行地存取数据,但这样就会产生多个用户程序并发存取同一数据的情况。若对并发操作不加控制,就可能会存取和存储不正确的数据,破坏数据库的一致性,所以数据库管理系统必须提供并发控制机制。

并发控制机制的好坏是衡量一个数据库管理系统性能的重要标志之一。SQL Server 以事务为单位,通常使用锁来实现并发控制。当用户对数据库并发访问时,为了确保事务完整性和数据库一致性,需要使用锁定。这样就可以保证任何时候都可以有多个正在运行的用户程序,但是所有用户程序都在彼此完全隔离的环境中运行。

本章主要介绍 SQL Server 2008 数据库系统的事务和锁的基本概念,事务和锁的分类及使用,以及通过锁的机制实现事务的并发控制。

8.1　事务概述

事务处理是数据库的主要工作,事务由一系列的数据操作组成,是数据库应用程序的基本逻辑单元,用来保证数据的一致性。SQL Server 提供了几种自动的可以通过编程来完成的机制,包括事务日志、SQL 事务控制语句,以及事务处理运行过程中通过锁定保证数据完整性的机制。

事务和存储过程类似,由一系列 T-SQL 语句组成,是 SQL Server 系统的执行单元。在数据库处理数据的时候,有一些操作是不可分割的整体。例如,当用银行卡消费的时候,首先要在账户扣除资金,然后再添加资金到商家的账户上。在这个过程中,用户所进行的实际

操作可以理解成是不可分割的，不能只扣除不添加，当然也不能只添加不扣除。

利用事务可以解决上面的问题，即把这些操作放在一个容器里，强制用户或者执行完所有的操作或者不执行任何一条语句。事务就是作为单个逻辑工作单元执行的一系列操作，这一系列的操作或者都被执行或者都不被执行。

在 SQL Server 2008 中，事务要求处理时必须满足四个原则，即原子性、一致性、隔离性和持久性。

(1) 原子性：事务必须是原子工作单元，对于其数据修改，要么全都执行，要么全都不执行。这一性质即使在系统崩溃之后仍能得到保证，在系统崩溃之后将进行数据库恢复，用来恢复和撤销系统崩溃时处于活动状态的事务对数据库的影响，从而保证事务的原子性。系统对磁盘上的任何实际数据修改之前都会将这个操作的日志记录到磁盘上。当发生崩溃时，系统能根据日志记录的当时该事务处于的状态，确定是撤销该事务所做的操作，还是将操作提交。

(2) 一致性：一致性要求事务执行完成后，将数据库从一个一致状态转变到另一个一致状态。即在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性，事务结束时，所有的内部数据结构都必须是正确的。例如在转账的操作中，各账户金额必须平衡，这一条规则对于程序员而言是一个强制的规定。

(3) 隔离性：也称为独立性，是指并行事务的修改必须与其他并行事务的修改相互独立。保证事务查看数据时数据所处的状态，只能是另一并发事务修改它之前的状态或者是修改它之后的状态，而不能是中间状态。隔离性意味着一个事务的执行不能被其他事务干扰，即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

(4) 持久性：在事务完成提交之后，就会对系统产生持久的影响，即事务的操作将写入数据库中，无论发生何种机器和系统故障都不应该对其有任何影响。例如，自动柜员机(ATM)在向客户支付一笔钱时，就不用担心丢失客户的取款记录。事务的持久性保证事务对数据库的影响是持久的，即使系统崩溃。

事务的这种机制保证了一个事务或者成功提交，或者失败回滚，二者必居其一，因此，事务对数据的修改具有可恢复性，即当事务失败时，它对数据的修改都会恢复到该事务执行前的状态。而使用一般的批处理，则有可能出现有的语句被执行，而另一些语句没有被执行的情况，从而有可能造成数据不一致。

8.2 事务的类型

根据事务的系统设置和运行模式的不同，SQL Server 将事务分为多种类型。

8.2.1 根据系统的设置分类

根据系统的设置，SQL Server 将事务分为两种类型：系统事务和用户定义事务。

1. 系统事务

系统事务是指在执行某些语句时，一条语句就是一个事务。但是要明确，一条语句的对象既可能是表中的一行数据，也可能是表中的多行数据，甚至是表中的全部数据。因此，只

有一条语句构成的事务也可能包含了对多行数据的处理。

CREATE、ALTER、DROP、INSERT、UPDATE、DELETE、SELECT、FETCH、OPEN、GRANT、REVOKE、TRUNCATE TABLE 等，这些语句本身就构成了一个事务。

【例 8-1】 使用 CREATE TABLE 创建一个表。

```
CREATE TABLE student
(  Id     CHAR(10),
   Name   CHAR(6),
   Sex    CHAR(2)
)
```

这条语句本身就构成了一个事务。

由于没有使用条件限制，那么这条语句就是创建包含 3 个列的表。要么全部创建成功，要么全部失败。

2. 用户定义事务

在实际应用中，大多数的事务都是用户定义的事务。在开发应用程序时，可以使用 BEGIN TRANSACTION 语句来定义明确的用户定义事务。在使用用户定义的事务时，一定要注意事务必须有明确的结束语句来结束。如果不使用明确的结束语句来结束，那么系统可能把从事务开始到用户关闭连接之间的全部操作都作为一个事务来对待。事务的明确结束可以使用两个语句中的一个：COMMIT TRANSACTION 语句或 ROLLBACK TRANSACTION 语句。COMMIT 是提交语句，将全部完成的语句明确地提交到数据库中。ROLLBACK 是回滚语句，该语句将事务的操作全部回滚，即表示事务操作失败。

还有一种特殊的用户定义事务，这就是分布式事务。如果事务是在一个服务器上的操作，其保证的数据完整性和一致性是指数据在一个服务器上的完整性和一致性。但是，如果是一个比较复杂的环境，可能有多台服务器，那么要保证在多台服务器环境中事务的完整性和一致性，就必须定义一个分布式事务。在这个分布式事务中，所有的操作都可以涉及多个服务器，当这些操作都成功时，那么所有这些操作都提交到相应服务器的数据库中；如果这些操作中有一个操作失败，那么这个分布式事务中的全部操作都将被回滚。

8.2.2 根据运行模式分类

根据运行模式的不同，SQL Server 将事务分为 4 种类型：自动提交事务、显式事务、隐式事务和批处理级事务。

1. 自动提交事务

如果没有通过任何语句设置事务，一条 T-SQL 语句就是一个事务，语句执行完事务就结束。以前我们使用的每一条 T-SQL 语句都可以叫作一个自动提交事务。

2. 显式事务

显式事务指每个事务均以 BEGIN TRANSACTION 语句、COMMIT TRANSACTION 或 ROLLBACK TRANSACTION 语句明确地定义了什么时候开始、什么时候结束。

3. 隐式事务

隐式事务指在前一个事务完成时新事务隐式开始，但每个事务仍以 COMMIT TRANSACTION 或 ROLLBACK TRANSACTION 语句显式结束。

4. 批处理级事务

批处理级事务是 SQL Server 2005 以后版本的新增功能，该事务只能应用于多个活动结果集(MARS)，在 MARS 会话中启动的 T-SQL 显式或隐式事务变为批处理级事务。当批处理完成时，没有提交或回滚的批处理级事务自动由 SQL Server 语句集合分组后形成单个的逻辑工作单元。

8.3 事务处理语句

所有的 T-SQL 语句本身都是内在的事务。另外，SQL Server 中有专门的事务处理语句，这些语句将 SQL 语句集合分组后形成单个的逻辑工作单元。事务处理的 T-SQL 语句包括：

(1) 定义一个事务的开始——BEGIN TRANSACTTCN；

(2) 提交一个事务——COMMIT TRANSACTION；

(3) 回滚事务——ROLLBACK TRANSACTION；

(4) 在事务内设置保存点——SAVE TRANSACTION。

BEGIN TRANSACTION 代表一个事务的开始点，每个事务继续执行直到用 COMMIT TRANSACTION 提交，从而正确地完成对数据库的永久改动；或者遇上错误用 ROLLBACK TRANSACTION 语句撤销所有改动，即回滚整个事务，也可以回滚到事务内的某个保存点，它也标志一个事务的结束。

1. BEGIN TRANSACTION 语句

BEGIN TRANSACTION 语句定义一个显式事务的起始点，即事务的开始。其语法格式为：

```
BEGIN { TRAN | TRANSACTION }
[ transaction_name | @tran_name_variable ]
[WITH MARK [ 'description']]
```

其中：

(1) TRANSACTION 关键字可以缩写为 TRAN。

(2) transaction_name 是给事务分配的名称，事务可以定义名称，也可以不定义名称，但是只能使用符合标识符规则的名字。

(3) @tran_name_variable 是含有效事务名称的变量名，必须用数据类型声明这个变量。

(4) WITH MARK 用于指定在日志中标记事务，description 是描述该标记的字符串。

2. COMMIT TRANSACTION 语句

COMMIT TRANSACTION 语句用于提交一个事务，标志一个成功的隐式事务或显式事务的结束。其语法格式为：

```
COMMIT [{ TRAN | TRANSACTION }
[ transaction_name | @tran_name_variable ] ]
```

对于 COMMIT TRANSACTION 语句需要注意以下几点：

(1) 因为数据已经永久修改,所以在 COMMIT TRANSACTION 语句后不能回滚事务。

(2) 在嵌套事务中使用 COMMIT TRANSACTION 时,内部事务的提交并不释放资源,也没有执行永久修改,只有在提交了外部事务时,数据修改才具有永久性,而且资源才会被释放。

3. ROLLBACK TRANSACTION 语句

ROLLBACK TRANSACTION 语句将显式事务或隐式事务回滚到事务的起点或事务内的某个保存点,它也标志一个事务的结束。其语法格式为:

```
ROLLBACK [ { TRAN | TRANSACTION }
[ transaction_name | @tran_name_variable
| savepoint_name | @savepoint_variable ] ]
```

对于 ROLLBACK TRANSACTION 语句需要注意以下几点:

(1) 如果不指定回滚的事务名称或保存点,则 ROLLBACK TRANSACTION 命令会将事务回滚到事务的起点。

(2) 在嵌套事务时,该语句将所有内层事务回滚到最远的 BEGIN TRANSACTION 语句,transaction_name 也只能是来自最远的 BEGIN TRANSACTION 语句的名称。

(3) 在执行 COMMIT TRANSACTION 语句后不能回滚事务。

(4) 如果在触发器中发出 ROLLBACK TRANSACTION 命令,将回滚对当前事务中所做的所有数据修改,包括触发器所做的修改。

(5) 事务在执行过程中出现任何错误,SQL Server 都将自动回滚事务。

4. SAVE TRANSACTION 语句

SAVE TRANSACTION 语句用于在事务内设置保存点。其语法格式为:

```
SAVE { TRAN | TRANSACTION }
 { savepoint_name | @savepoint_variable }
```

在事务内的某个位置建立一个保存点,使用户可以将事务回滚到该保存点的状态,而不回滚整个事务。

使用事务时应注意以下几点:

(1) 不是所有的 T-SQL 语句都能放在事务里,通常 INSERT、UPDATE、DELETE、SELECT 等可以放在事务里,而创建、删除、恢复数据库等则不能放在事务里。

(2) 事务要尽量小,而且一个事务占用的资源越少越好。

(3) 如果在事务中间发生了错误,并不是所有情况都会回滚,只有达到一定的错误级别才会回滚,可以在事务中使用@@Error 变量查看是否发生了错误。

【例 8-2】 定义一个事务,将所有选修了 C004 号课程的学生的分数加 5 分,并提交该事务。

```
DECLARE @t_name CHAR(10)
SET @t_name = 'add_score'
BEGIN TRANSACTION @t_name
USE teaching
UPDATE sc SET score = score + 5
```

```
WHERE cno = 'C004'
COMMIT TRANSACTION @t_name
```

【例 8-3】 定义一个事务，向 teaching 数据库的 student 表中插入一行数据，然后再删除该行。执行后，新插入的数据行并没有被删除。利用事务保存点来完成。

```
BEGIN TRANSACTION
USE teaching
INSERT INTO student(sno,sname,ssex,birthday,en_time,specialty,grade)
VALUES('201701001','朱一虹','女',1999-5-6,'2017-9-5','电子信息','2017 级')
SAVE TRAN savepoint
DELETE FROM student WHERE sname = '朱一虹'
ROLLBACK TRAN savepoint
COMMIT
```

【例 8-4】 定义一个事务，向 teaching 数据库的 student 表中插入一行数据，如果插入成功，则向 sc 表中插入一行或多行此学生的选课信息，并显示"添加成功!"；如果插入失败，则不向 sc 表中插入数据，并显示"添加失败!"。

```
BEGIN TRANSACTION
USE teaching
INSERT INTO student VALUES('201702001','李虹','女',1999-8-16,'2017-9-5','计算机','2017 级')
IF @@Error = 0
 BEGIN
   INSERT INTO sc (sno,cno)VALUES('201702001','C001')
   INSERT INTO sc (sno,cno)VALUES('201702001','C004')
   PRINT '添加成功!'
   COMMIT
 END
ELSE
  BEGIN
    PRINT '添加失败!'
    ROLLBACK
  END
```

【例 8-5】 定义一个转账事务，向 bankcard 数据库的 Trecord 表中插入一行某账号的转出记录数据和另一账号的转入记录数据。两个插入操作都会触发 Transactions 触发器，修改相应的余额。

```
BEGIN TRANSACTION
INSERT Trecord VALUES(getdate(),'41254280033512060000',500,NULL,'43674280033512050000',
NULL,'转账支出')
INSERT Trecord VALUES(getdate(),'43674280033512050000',NULL,500,'41254280033512060000',
NULL,'转账收入')
COMMIT
```

8.4 事务的并发控制

并发控制指的是当多个用户同时更新行时，用于保护数据库完整性的各种技术，目的是保证一个用户的工作不会对另一个用户的工作产生不合理的影响。在某些情况下，这些措

施保证了当用户和其他用户一起操作时，所得的结果和他单独操作时的结果是一样的。

8.4.1 并发带来的问题

并发性是指多个用户可以同时对同一数据进行操作。特别是对于网络数据库来说，这个特点更加突出。提高数据库的处理速度，单单依靠提高计算机的物理速度是不够的，还必须充分考虑数据库的并发性问题，提高数据库并发操作的效率。

当多个用户同时读取或修改相同的数据库资源的时候，通过并发控制机制可以控制用户的读取和修改。如果多个用户同时访问一个数据库且没有加以控制，则当他们的事务同时使用相同的数据时就可能会发生问题，这些问题包括以下几种情况。

(1) 丢失修改：指在一个事务读取一个数据时，另外一个事务也访问同一数据。那么，在第一个事务中修改了这个数据后，第二个事务也修改了这个数据。这样第一个事务内的修改结果就被丢失，因此称为丢失修改。

例如，事务 T1 读取某表中数据 A＝20，事务 T2 也读取 A＝20，事务 T1 修改 A＝A－1，事务 T2 也修改 A＝A－1；最终结果 A＝19，事务 T1 的修改丢失了。

(2) 脏读：指当一个事务正在访问数据，并且对数据进行了修改，而这种修改还没有提交到数据库中时，另外一个事务也访问这个数据，然后使用了这个数据。因为这个数据是还没有提交的数据，那么另外一个事务读到的这个数据就是“脏数据”，依据“脏数据”所做的操作可能是不正确的。

例如，事务 T1 读取某表中数据 A＝20，并修改 A＝A－1，写回数据库，此时事务 T2 读取 A＝19，但事务 T1 回滚了前面的操作，而事务 T2 也修改 A＝A－1，导致最终 A＝18。事务 T2 读取的就是“脏数据”。

(3) 不可重复读：指在一个事务内，多次读同一数据。在这个事务还没有结束时，另外一个事务也访问同一数据。那么，在第一个事务中的两次读数据之间，由于第二个事务的修改，造成第一个事务两次读到的数据可能是不一样的。在一个事务内两次读到的数据不一样，就称为不可重复读。

例如，事务 T1 读取某表中数据 A＝20，B＝30，求 C＝A＋B，得到 C＝50，事务 T1 继续往下执行；事务 T2 读取 A＝20，修改 A＝A * 2，A＝40；事务 T1 又一次读取数据 A＝40、B＝30，求 C＝A＋B，得到 C＝70；所以，在事务 T1 内两次读到的数据是不一样的，即不可重复读。

(4) 幻读：与不可重复读相似，是指当事务不是独立执行时发生的一种现象。例如，第一个事务对一个表中的全部数据行都进行了某种修改；同时，第二个事务向表中插入了一行数据。那么，以后就会发生操作第一个事务的用户发现表中还有没有修改的数据行，就好像出现了幻觉一样。当对某条记录执行插入或删除操作而该记录属于某个事务正在读取的行的范围时，也会发生幻读问题。

为防止出现上述数据不一致的情况，必须使并发的事务串行化，使各事务都按照某种次序来进行，从而消除相互干扰，这种机制就是锁。

8.4.2 锁的基本概念

锁是实现并发控制的主要方法，是防止其他事务访问指定的资源、实现并发控制的一种

手段，是多个用户能够同时操纵同一个数据库中的数据而不发生数据不一致现象的重要保障。

为了提高系统的性能、加快事务的处理速度、缩短事务的等待时间，应该使锁定的资源最小化。为了控制锁定的资源，应该首先了解系统的空间管理。在 SQL Server 2008 中，最小空间管理单位是页，一个页有 8KB。所有的数据、日志、索引都存放在页上。另外，使用页有一个限制，这就是表中的一行数据必须在同一个页上，不能跨页。页上面的空间管理单位是簇，一个簇是 8 个连续的页。表和索引的最小占用单位是簇。数据库由一个或多个表或者索引组成，即由多个簇组成。SQL Server 系统的空间管理结构示意图如图 8-1 所示。

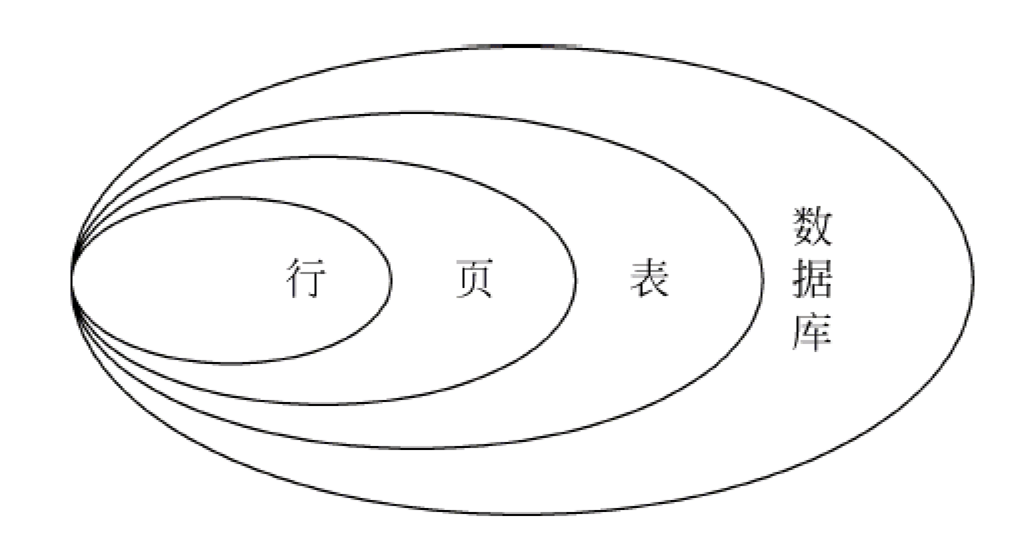


图 8-1　SQL Server 空间管理

数据库中的锁是一种软件机制，用来指示某个用户已经占用了某种资源，从而防止其他用户做出影响该用户或会导致数据库数据的不完整性和不一致性的数据修改。这里所谓的资源，主要指用户可以操作的数据行、索引以及数据表等。根据资源的不同，锁有多粒度的概念，也就是指可以锁定的资源的层次。SQL Server 中能够锁定的资源粒度主要包括数据库、表、页、行标识符(即表中的单行数据)等等。

多粒度锁的重要用途是支持并发操作和保证数据的完整性。SQL Server 根据用户的请求，做出分析后自动给数据库加上合适的锁。假设某用户只操作一个表中的部分行数据，系统可能会只添加几个行锁或页面锁，这样可以尽可能多地支持多用户的并发操作。但是，如果用户事务中频繁对某个表中的多条记录操作，将导致对该表的许多记录行都加上行锁，数据库系统中锁的数目会急剧增加，这样就加重了系统负荷，影响系统性能。因此，在数据库系统中，一般都支持锁升级。所谓锁升级，是指调整锁的粒度，将多个低粒度的锁替换成少数更高粒度的锁，以此来降低系统负荷。在 SQL Server 中，当一个事务中的锁较多，达到锁升级门限时，系统自动将行级锁和页面锁升级为表级锁。特别值得注意的是，在 SQL Server 中，锁的升级门限以及锁升级是由系统自动来确定的，不需要用户设置。

8.4.3　锁的类型

数据库引擎使用不同类型的锁锁定资源，这些锁确定了并发事务访问资源的方式。

SQL Server 2008 中常见的锁有以下几种。

1. 共享锁

共享锁(Shared Lock，S 锁)允许并发事务读取(SELECT)一个资源。资源上存在 S 锁时，任何其他事务都不能修改数据。一旦已经读取数据，便立即释放资源上的 S 锁，除非在事务生存周期内用锁定提示保留 S 锁。

2. 排他锁

排他锁(Exclusive Lock,X 锁)可以防止并发事务对资源进行访问,其他事务不能读取或修改 X 锁锁定的数据。即 X 锁锁定的资源只允许进行锁定操作的程序使用,其他任何对它的操作均不会被接受。执行数据更新命令即 INSERT、UPDATE 或 DELETE 命令时 SQL Server 会自动使用 X 锁,但当对象上有其他锁存在时无法对其加 X 锁。X 锁一直到事务结束才能被释放。

3. 更新锁

更新锁(Update Lock,U 锁)可以防止通常形式的死锁。一般更新模式由一个事务组成,此事务读取记录,获取资源(页或行)的 S 锁,然后修改行,此操作要求锁转换为 X 锁。如果两个事务获得了资源上的 S 锁,然后试图同时更新数据,则一个事务尝试将锁转换为 X 锁。S 锁到 X 锁的转换必须等待一段时间,因为一个事务的 X 锁与其他事务的 S 锁不兼容,此时发生锁等待,而第二个事务也试图获取 X 锁以进行更新;由于两个事务都要转换为 X 锁,并且每个事务都等待另一个事务释放 S 锁,因此发生死锁。

U 锁就是为了防止这种死锁而设立的。当 SQL Server 准备更新数据时,它首先对数据对象加 U 锁,锁定的数据将不能被修改,但可以读取,所以 U 锁可以与 S 锁共存。等到 SQL Server 确定要进行更新数据操作时,它会自动将 U 锁换为 X 锁,但当数据对象上有其他 U 锁存在时则无法对其作 U 锁锁定。

4. 意向锁

如果对一个资源加意向锁(Intent Lock,I 锁),则说明该资源的下层资源正在被加锁(S 锁或 X 锁);对任一资源加锁时,必须先对它的上层资源加 I 锁。

系统使用 I 锁来最小化锁之间的冲突。I 锁建立一个锁机制的分层结构,这种结构依据锁定的资源范围从低到高依次是行级锁、页级锁和表级锁。I 锁表示系统希望在层次低的资源上获得 S 锁或者 X 锁。

例如,放置在表级上的 I 锁表示一个事务可以在表中的页或者行上放置 S 锁。I 锁可以提高性能,这是因为系统只需要在表级上检查 I 锁,就能确定一个事务能否在那个表上安全地获取一个锁,而不需要检查表上的每一个行锁或者页锁后再确定一个事务是否可以锁定整个表。

常用的 I 锁有 3 类:意向共享锁(IS 锁);意向排他锁(IX 锁);共享意向排他锁(SIX 锁)。

(1) IS 锁:IS 锁表示某事务有读取低层次资源的意向,把 S 锁放在这些低层次的单个资源上。也就是说,如果对一个数据对象加 IS 锁,表示它的后裔资源拟(意向)加 S 锁。例如,要对某个元组加 S 锁,则要首先对包含该元组的关系和数据库加 IS 锁。

(2) IX 锁:IX 锁表示某事务有修改低层次资源的意向,把 X 锁放在这些低层次的单个资源上。也就是说,如果对一个数据对象加 IX 锁,表示它的后裔资源拟(意向)加 X 锁。例如,要对某个元组加 X 锁,则要首先对包含该元组的关系和数据库加 IX 锁。

(3) SIX 锁:SIX 锁是 S 锁和 IX 锁的组合。使用 SIX 锁表示某事务有读取顶层资源和修改一些低层次资源的意向,把 IX 锁放在这些低层次单个资源上。也就是说,如果对一个数据对象加 SIX 锁,表示对它加 S 锁,再加 IX 锁,即 SIX=S+IX。例如,对某个表加 SIX 锁,则表示该事务要读整个表(所以要对该表加 S 锁),同时会更新个别元组(所以要对该表加 IX 锁)。

5. 模式锁

模式锁(Schema Lock)保证当表或者索引被另外一个会话使用时,其结构模式不能被删除或修改。SQL Server 系统提供了两种类型的模式锁:模式稳定锁(Sch-S)和模式修改锁(Sch-M)。模式稳定锁确保锁定的资源不能被删除,模式修改锁确保其他会话不能使用正在修改的资源。

执行表的数据定义语言操作(例如,添加列或删除表等)时使用模式修改锁。当编译查询时,使用模式稳定锁。模式稳定锁不阻塞任何事务锁,包括 X 锁。因此在编译查询时,其他事务(包括在表上有 X 锁的事务)都能继续运行,但不能在表上执行 DDL 操作。

6. 大容量更新锁

当将数据大容量复制到表,且指定了 TABLOCK 提示或者使用 sp_tableoption 设置了 table_lock_on_bulk 表选项时,将使用大容量更新锁(Bulk Update Lock)。大容量更新锁允许事务将数据并发地大容量复制到同一个表中,同时防止其他不进行大容量复制数据的事务访问该表。

8.4.4 锁的信息

1. 锁的兼容性

在一个事务已经对某个对象锁定的情况下,另一个事务请求对同一个对象的锁定,此时就会出现锁定兼容性问题。当两种锁定方式兼容时,可以同意对该对象的第二个锁定请求。如果请求的锁定方式与已挂起的锁定方式不兼容,那么就不能同意第二个锁定请求。相反,第二个请求要等到第一个事务释放其锁定,并且释放所有其他现有的不兼容锁定为止。

例如,当第一个事务控制排他锁时,在第一个事务结束并释放排他锁之前,其他事务不能在该资源上获取任何类型的(共享、更新或排他)锁。另一种情况下,如果共享锁已应用到资源,其他事务还可以获取该项目的共享锁或更新锁,即使第一个事务尚未完成。但是,在释放共享锁之前,其他事务不能获取排他锁。

资源锁模式有一个兼容性矩阵,显示了与在同一资源上可获取的其他锁相兼容的锁,见表 8-1。

表 8-1 锁的兼容性

锁 A	锁 B					
	IS	S	IX	SIX	U	X
IS	是	是	是	是	是	否
S	是	是	否	否	是	否
IX	是	否	是	否	否	否
SIX	是	否	否	否	否	否
U	是	是	否	否	否	否
X	否	否	否	否	否	否

关于锁的兼容性的一些说明:

(1) IX 锁与 IX 锁模式兼容,因为 IX 锁表示打算更新一些行而不是所有行,因此允许其他事务读取或更新另一部分行,只要这些行之间不冲突即可。

(2) 模式稳定锁与除了模式修改锁之外的所有锁都兼容。

(3) 模式修改锁与其他所有锁都不兼容。

(4) 大容量更新锁只与模式稳定性及其他大容量更新锁相兼容。

2. 查看锁的信息

在 SQL Server 2008 中，一般可以使用 SSMS 中的对象资源管理器浏览系统中的锁，也可以使用系统存储过程 SP_LOCK 或查询系统表 sys. dm_tran_locks 来查看锁的信息。

(1) 进入 SSMS 的“对象资源管理器”面板，右击服务器名，选择“活动和监视器”命令，如图 8-2 所示。



图 8-2　在“对象资源管理器”中查看锁的信息

(2) 选择“进程”选项，可以看到锁的具体信息。如图 8-3 所示为在“进程”中查看锁的具体信息。



图 8-3　在“进程”中查看锁的信息

8.4.5　死锁的产生及解决办法

锁机制的引入能解决并发用户的数据不一致性问题，但也会引起事务间的死锁问题。

在事务和锁的使用过程中,死锁是一个不可避免的现象。在数据库系统中,死锁是指多个用户分别锁定了一个资源,并又试图请求锁定对方已经锁定的资源,这就产生了一个锁定请求环,导致多个用户都处于等待对方释放所锁定资源的状态。通常,根据用户的操作需求使用不同的锁类型锁定资源,然而当某组资源的两个或多个事务之间有循环相关性时,就会发生死锁现象。

产生死锁的情况一般包括以下两种:

第一种情况,当两个事务分别锁定了两个单独的对象,这时每一个事务都要求在另外一个事务锁定的对象上获得一个锁,因此每一个事务都必须等待另外一个事务释放占有的锁,这时,就发生了死锁。这种死锁是最典型的死锁形式。

第二种情况,当在一个数据库中,有若干个长时间运行的事务执行并行的操作,当查询分析器处理一种非常复杂的查询(例如连接查询)时,那么由于不能控制处理的顺序,有可能发生死锁现象。

在数据库中解决死锁常用的方法有如下几种。

(1) 要求每个事务一次就将要使用的数据全部加锁,否则就不能继续执行;或者,预先规定一个顺序,所有事务都按这个顺序加锁,这样就不会发生死锁。

(2) 允许死锁发生,系统采用某些方式诊断当前系统中是否有死锁发生。在 SQL Server 中,系统能够自动定期搜索和处理死锁问题。系统在每次搜索中标识所有等待锁定请求的事务,如果在下一次搜索中该被标识的事务仍处于等待状态,SQL Server 就开始递归死锁搜索。当搜索检测到锁定请求环时,系统将根据事务的死锁优先级别来结束一个优先级最低的事务,此后,系统回滚该事务,并向该进程发出 1205 号错误信息。这样,其他事务就有可能继续运行了。

死锁优先级的设置语句为:SET DEADLOCK_PRIORITY { LOW | NORMAL}。其中,LOW 说明该进程会话的优先级较低,在出现死锁时,可以首先中断该进程的事务。另外,通过设置 LOCK_TIMEOUT 选项能够设置事务处于锁定请求状态的最长等待时间。该设置的语句为:SET LOCK_TIMEOUT { timeout_period }。其中,timeout_period 以毫秒为单位。

8.4.6 手工加锁

建议让 SQL Server 系统自动管理锁,该系统会分析用户的 SQL 语句要求,自动为该请求加上合适的锁,而且在锁的数目太多时,系统会自动进行锁升级。如前所述,升级的门限由系统自动配置,并不需要用户配置。

在实际应用中,有时为了应用程序正确运行和保持数据的一致性,必须人为地给数据库的某个表加锁。比如,在某应用程序的一个事务操作中,需要根据一个编号对几个数据表做统计操作,为保证统计数据时间的一致性和正确性,从统计第一个表开始到全部表结束,其他应用程序或事务不能再对这几个表写入数据,这个时候,该应用程序希望在从统计第一个数据表开始或在整个事务开始时能够由程序人为地(显式地)锁定这几个表,这就需要用到手工加锁(也称显式加锁)技术。

SQL Server 的 SELECT、INSERT、DELETE、UPDATE 语句支持显式加锁。这 4 个语句在显式加锁的语法上类似,下面仅以 SELECT 语句为例给出语法:

```
SELECT FROM [ WITH ]
```

其中,[WITH]指需要在该语句执行时添加在该表上的锁类型,所指定的锁类型有如下几种:

(1) HOLDLOCK——在该表上保持共享锁,直到整个事务结束,而不是在语句执行完立即释放所添加的锁。

(2) NOLOCK——不添加共享锁和排他锁,当这个选项生效后,可能会读到未提交读的数据或“脏数据”,这个选项仅仅应用于 SELECT 语句。

(3) PAGLOCK——指定添加页面锁(否则通常可能添加表锁)。

(4) READCOMMITTED——设置事务为读提交隔离性级别。

(5) READPAST——跳过已经加锁的数据行,这个选项将使事务读取数据时跳过那些已经被其他事务锁定的数据行,而不是阻塞直到其他事务释放锁。READPAST 仅仅应用于 READCOMMITTED 隔离性级别下事务操作中的 SELECT 语句操作。

(6) EADUNCOMMITTED——等同于 NOLOCK。

(7) REPEATABLEREAD——设置事务为可重复读隔离性级别。

(8) ROWLOCK——指定使用行级锁。

(9) SERIALIZABLE——设置事务为可串行的隔离性级别。

(10) TABLOCK——指定使用表级锁,而不是使用行级或页面级的锁,SQL Server 在该语句执行完后释放这个锁,而如果同时指定了 HOLDLOCK,则该锁一直保持到这个事务结束。

(11) TABLOCKX——指定在表上使用排他锁,这个锁可以阻止其他事务读或更新这个表的数据,直到这个语句或整个事务结束。

(12) UPDLOCK——指定在读表中数据时设置修改锁(UPDATE LOCK,U 锁)而不是设置共享锁,该锁一直保持到这个语句或整个事务结束。使用 UPDLOCK 的作用是允许用户先读取数据(而且不阻塞其他用户读数据),并且保证在后来再更新数据时,这一段时间内这些数据没有被其他用户修改。

【例 8-6】 系统自动加排他锁的情况。

新建两个连接,在第一个连接中执行以下语句:

```
BEGIN TRAN
UPDATE   student SET sname = '王一'   WHERE sno = '201502001'
WAITFOR DELAY   '00:00:30'      -- 等待 30 秒
COMMIT TRAN
```

在第二个连接中执行以下语句:

```
SELECT   *   FROM student WHERE   sno = '201502001'
```

若同时执行上述两个语句,则 SELECT 查询必须等待 UPDATE(系统自动加排他锁)执行完毕才能执行,即要等待 30 秒。

【例 8-7】 人为加 HOLDLOCK 锁的情况。

新建三个连接,在第一个连接中执行以下语句:

```
BEGIN TRAN
```

```
 SELECT  *  FROM  student  (HOLDLOCK)  --HOLDLOCK 人为加锁
WHERE sno='201502001'
WAITFOR  DELAY  '00:00:30'     --等待 30 秒
COMMIT TRAN
```

在第二个连接中执行以下语句：

```
SELECT  *  FROM student WHERE  sno='201502001'
```

在第三个连接中执行以下语句：

```
UPDATE student SET  sname='张明明'  WHERE  sno='201502001'
```

若同时执行上述三个语句，则第二个连接中的 SELECT 查询可以执行，而第三个连接中 UPDATE 必须等待第一个连接中的共享锁结束后才能执行，即要等待 30 秒。

由上可见，在 SQL Server 中可以灵活多样地为 SQL 语句显式加锁，若适当使用，完全可以完成一些程序的特殊要求，保证数据的一致性和完整性。对于一般使用者而言，了解锁机制并不意味着必须使用它。事实上，SQL Server 建议让系统自动管理数据库中的锁，而且一些关于锁的设置选项也没有提供给用户和数据库管理人员。对于特殊用户，通过给数据库中的资源显式加锁，可以满足很高的数据一致性和可靠性要求，只是需要特别注意避免死锁现象的出现。

习　题　8

1. 什么是事务？如果要提交或取消一个事务，可使用什么语句？
2. 事务分为哪几类？
3. 简述事务回滚机制。
4. 简述锁机制。锁分为哪几类？
5. 分析各类锁之间的兼容性。
6. 简述死锁及其解决办法。
7. 创建一个事务，将所有女生的考试成绩都加 5 分，将所有男生的考试成绩都减 5 分，并提交。
8. 创建一个事务，向 bankcard 数据库的 depositor 表中添加一条记录，设置保存点；再向 account 表中添加一条记录，如果在插入第二条语句时出现错误，则回滚到保存点。

第二部分
数据库安全

第 9 章　数据库访问的安全性管理

安全性对于任何数据库管理系统来说都是至关重要的,数据库访问的安全性是指保护数据库以防止因不合法用户的访问而造成数据的泄密或破坏。SQL Server 2008 提供有效的数据访问安全机制,在数据库管理系统中,用检查口令等手段来检查用户身份,从而保证只有合法的用户才能进入数据库系统。当用户对数据库执行操作时,系统自动检查用户是否有权限进行这些操作。

对于系统管理员、数据库编程人员,甚至对于每个用户来说,数据库系统的安全性都是至关重要的。本章主要介绍 SQL Server 2008 的两种数据库身份验证模式及其设置,服务器登录账号的创建方法,数据库用户的创建方法,角色和权限设置、管理和使用等。

9.1　身 份 验 证

当用户使用 SQL Server 2008 时,需要经过两个安全性阶段：身份验证阶段和权限认证阶段。

在身份验证阶段,用户在 SQL Server 上获得对任何数据库的访问权限之前,必须登录到 SQL Server 2008 上,并且被认为是合法的。SQL Server 2008 或者 Windows 对用户进行验证,如果验证通过,用户就可以连接到 SQL Server 2008 服务器上；否则,服务器将拒绝用户登录,从而保证了系统的安全性。

用户验证通过,登录到 SQL Server 2008 服务器上之后,还需要检测用户是否有访问服务器中数据的权限,为此需要将登录账号映射为某些数据库的用户,并为数据用户授予访问权限,权限认证可以控制用户对数据库进行操作。

9.1.1　SQL Server 的身份验证模式

身份验证模式用来确认登录 SQL Server 用户的登录账号和密码的正确性,验证其是否具有连接 SQL Server 的权限。

在身份验证阶段, SQL Server 和 Windows 是组合在一起的,因此 SQL Server 提供了两种确认用户的验证模式：Windows 验证模式和混合验证模式。

1. Windows 验证模式

SQL Server 2008 数据库系统通常运行在 Windows 服务器平台上,Windows 本身具备管理登录、验证用户合法性的能力。SQL Server 可以使用 Windows 操作系统的安全机制来验证用户身份,在这种模式下,只要用户能够通过 Windows 的用户身份验证,即可连接到 SQL Server 2008 服务器上,而 SQL Server 本身不需要管理一套登录数据。

在 Windows 验证模式下,SQL Server 检测当前使用 Windows 的用户账户,并在系统注册表中查找该用户,以确定该用户是否有权限登录。这种验证模式只适用于能够有效进行身份验证的 Windows 操作系统,在其他的操作系统下无法使用。SQL Server 的登录安全性直接集成到 Windows 的安全特性上,例如安全验证和密码加密、审核、密码过期、最短密码长度,以及在多次登录请求无效后锁定账号。

Windows 验证模式具有如下优点:

(1) 数据库管理员的工作可以集中在管理数据库上,而不是管理用户账户,对用户账户的管理可以交给 Windows 去完成。

(2) Windows 有着更强的用户账户管理工具,可以设置账户锁定、密码期限等,如果不是通过定制来扩展 SQL Server,那么 SQL Server 是不具备这些功能的。

(3) Windows 的组策略支持多个用户同时被授权访问 SQL Server。

2. 混合验证模式

混合身份验证模式使用户可以使用 Windows 身份验证或 SQL Server 身份验证与 SQL Server 2008 服务器连接。它将区分用户账号在 Windows 操作系统下是否可信,对于可信的连接用户系统,直接采用 Windows 身份验证模式;否则,SQL Server 2008 会通过账户的存在性和密码的匹配性自行进行验证。例如,允许某些非可信的 Windows 用户连接 SQL Server 2008 服务器,它通过检查是否已设置 SQL Server 2008 登录账户以及输入的密码是否与设置的密码相符来进行验证。如果 SQL Server 2008 服务器未设置登录信息,则身份验证失败,而且用户会收到错误提示信息。

在混合验证模式下,使用哪个模式取决于最初通信时使用的网络库。如果一个用户使用 TCP/IP SOCKETS 进行登录验证,则将使用 SQL Server 验证模式;如果用户使用命名管道,则登录时使用 Windows 验证。在 SQL Server 验证模式下,处理登录的过程为在输入登录名和密码后,SQL Server 在系统注册表中检测输入的登录名和密码,如果输入的登录名和密码正确,就可以登录到 SQL Server 服务器上。

混合验证模式具有如下优点:

(1) 创建了 Windows 之上的另外一个安全层次。

(2) 支持更大范围的用户,例如非 Windows 客户、Novell 网络等。

(3) 一个应用程序可利用单个的 SQL Server 登录账户或密码。

9.1.2 设置身份验证模式

在第一次安装 SQL Server 或者使用 SQL Server 连接其他服务器时,需要指定验证模式。对于已经指定验证模式的 SQL Server 服务器,在 SQL Server 中还可以进行修改。SQL Server 的安全系统必须保证不能被未通过验证的用户访问。

在 SSMS 中设置身份验证模式的基本步骤如下。

(1) 打开 SSMS,在对象资源管理器窗口中的目标服务器上右击,弹出快捷菜单,选择“属性”命令,如图 9-1 所示。

(2) 出现“服务器属性”窗口,选择“选择页”中的“安全性”选项,进入安全性设置页面,如图 9-2 所示。

(3) 在“服务器身份验证”选项组中选择验证模式前的单选按钮,选中需要的验证模式。



图 9-1　利用对象资源管理器设置身份验证模式

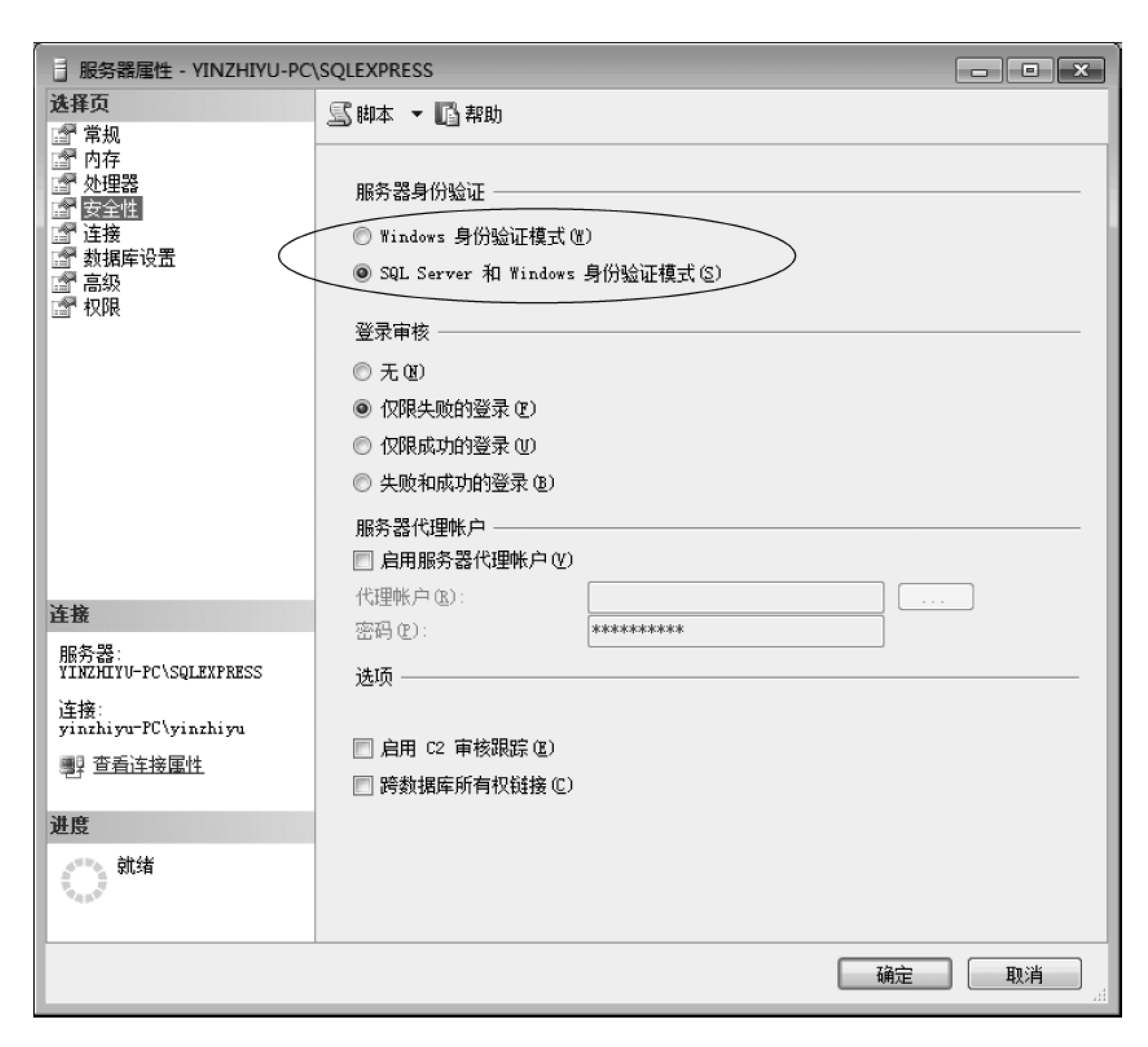


图 9-2　“服务器属性”窗口的“安全性”页面

还可以在“登录审核”选项组中设置需要的审核方式。审核方式取决于安全性要求，这 4 种审核级别的含义如下：

① 无——不使用登录审核。

② 仅限失败的登录——记录所有的失败登录。

③ 仅限成功的登录——记录所有的成功登录。

④ 失败和成功的登录——记录所有的登录。

(4) 单击“确定”按钮,出现如图 9-3 所示的对话框,重启 SQL Server 完成服务器登录验证模式的设置。



图 9-3　设置服务器身份验证模式提示

9.2 账号管理

Windows 用户账号和 SQL Server 登录账号允许用户登录到 SQL Server 系统中。如果用户想继续对系统中的某个特定数据库进行操作,就必须有一个数据库用户账号。每个数据库要求单独的用户账户,每个用户账户都拥有该数据库中对象(表、视图和存储过程等)应用的一些安全权限,用户在数据库中进行的所有活动由 T-SQL 语句传到 SQL Server 的服务器上,以确定是否有权限。

所以,对于每一个要使用的数据库,用户必须拥有该数据库的账号。当然,如果没有这些特定的账号,用户也可以用 guest 登录。数据库用户账号可以从已经存在的 Windows 用户账号、Windows 用户组、SQL Server 的登录名或者角色映射过来。

9.2.1 服务器登录账号

登录属于服务器级的安全策略,要连接到数据库,首先要存在一个合法的登录账号。

1. 创建服务器登录账号

在 SSMS 中创建服务器登录账号的步骤如下:

(1) 在 SSMS 的“对象资源管理器”窗口中展开“安全性”选项。在“登录名”上右击,在弹出快捷菜单中选择“新建登录名”命令,如图 9-4 所示。



图 9-4　在“对象资源管理器”中创建登录账号

(2) 在“登录名”对话框中,首先选择登录的验证模式,选中其前面的单选按钮,如图 9-5 所示。如果选中了“Windows 身份验证”,则“登录名”设置为 Windows 登录账号即可,无须设置密码;如果选中了“SQL Server 身份验证”,则需要设置一个“登录名”以及“密码”和“确认密码”。最后可以再进行其他参数的设置。

(3) 进入“登录名-新建”对话框,如图 9-5 所示,选择“选择页”中的“服务器角色”选项,出现服务器角色设定页面,如图 9-6 所示,可以为此登录账号的用户添加服务

器角色，当然也可以不为此用户添加任何服务器角色。



图 9-5　设置登录账号对话框

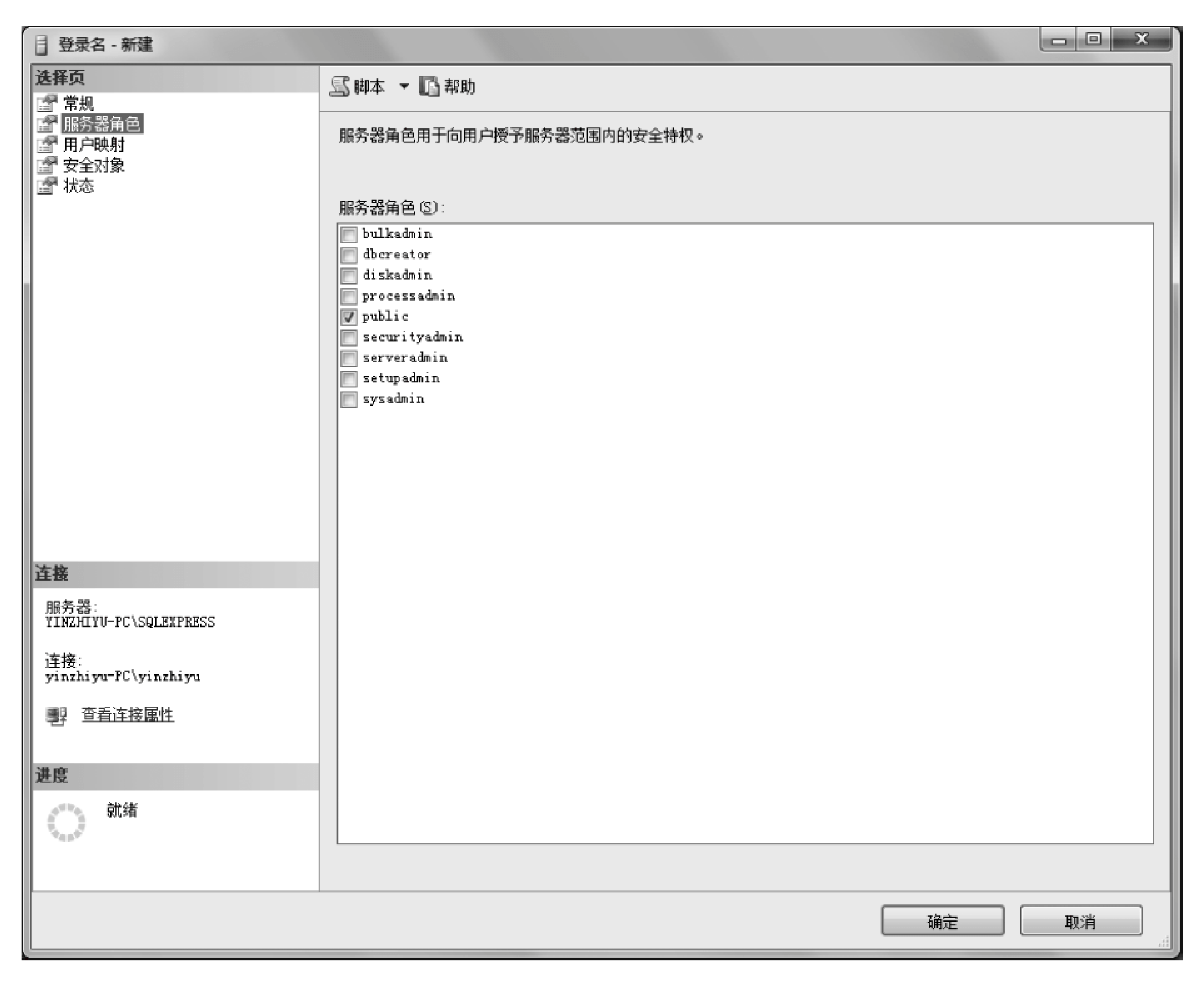


图 9-6　为登录账号添加服务器角色

(4) 在“登录名-新建”对话框中，选择“选择页”中的“用户映射”选项，进入映射设置页面，可以为这个新建的登录添加映射到此登录名的用户，并添加数据库角色，从而使该用户获得数据库的相应角色对应的数据库权限，如图 9-7 所示。同样也可以不为此用户添加任何数据库角色。

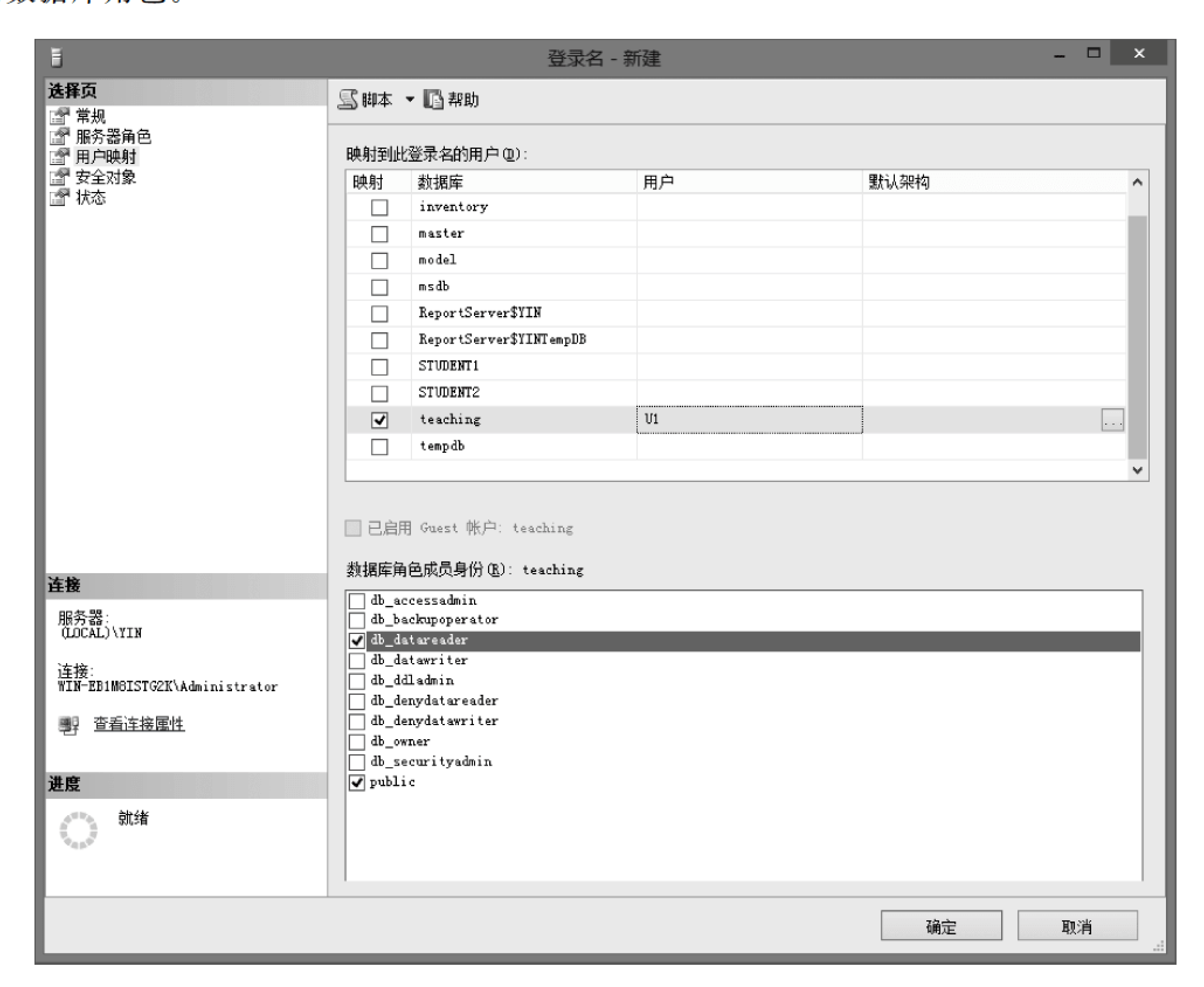


图 9-7　为登录账号添加数据库角色

(5) 单击“确定”按钮，服务器登录账号创建完毕。

2. 查看服务器登录账号

可以使用“对象资源管理器”查看登录账号：在 Microsoft SQL Server Management Studio 中进入“对象资源管理器”面板，展开“安全性”选项，再展开“登录名”选项，即可看到系统创建的默认登录账号以及建立的其他登录账号，如图 9-8 所示。

3. T-SQL 语句创建服务器登录账号

T-SQL 语句创建服务器登录账号的语法格式如下：

```
CREATE LOGIN login_name { WITH <option_list1> | FROM <sources> }
<option_list1> ::=
    PASSWORD = { 'password' | hashed_password HASHED } [ MUST_CHANGE ]
    [ , <option_list2> [ ,... ] ]
<option_list2> ::=
    SID = sid
    | DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
```

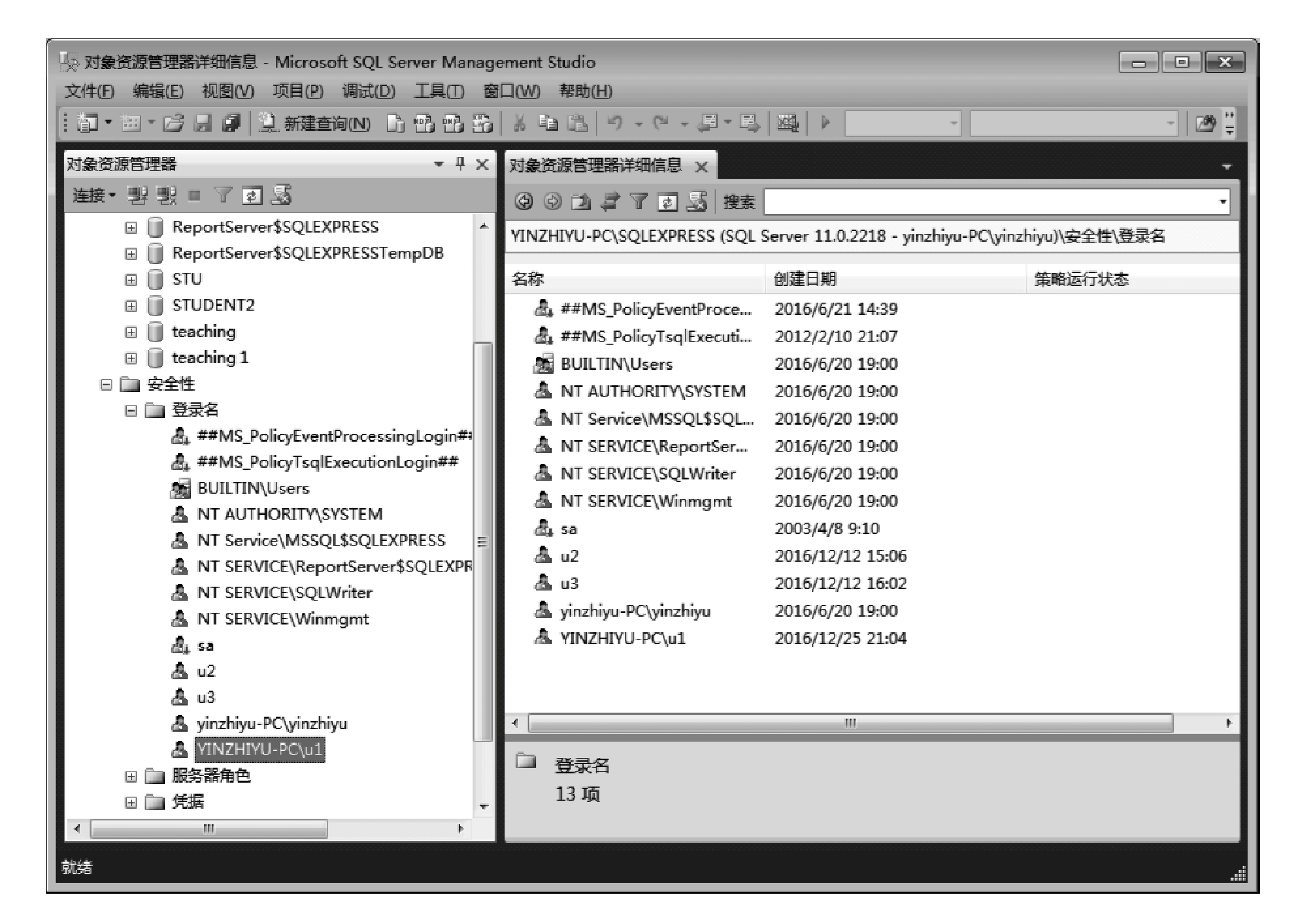


图 9-8　查看服务器登录账号

```
    | CHECK_EXPIRATION = { ON | OFF}
    | CHECK_POLICY = { ON | OFF}
    | CREDENTIAL = credential_name
< sources > : : =
    WINDOWS [ WITH < windows_options >[ ,… ] ]
    | CERTIFICATE certname
    | ASYMMETRIC KEY asym_key_name
< windows_options > : : =
    DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
```

其中，比较重要的几个参数的说明如下：

(1) login_name——指定创建的登录名。有四种类型的登录名：SQL Server 登录名、Windows 登录名、证书映射登录名和非对称密钥映射登录名。如果创建的登录名从 Windows 账户映射而来，则必须使用[< domainName >\< login_name >]格式的登录名。SQL Server 身份验证登录必须是符合标识符命名规则的登录名。

(2) PASSWORD = 'password'——适用于仅限 SQL Server 登录名的登录。指定正在创建的登录名的密码。

(3) PASSWORD = hashed_password——仅适用于 HASHED 关键字。指定要创建的登录名对应密码的哈希值。

(4) MUST_CHANGE——适用于 SQL Server 仅限的登录。如果包括此选项，则 SQL Server 将在首次使用新登录名时提示用户输入新密码。

(5) DEFAULT_DATABASE = database——指定将指派给登录名的默认数据库。如果未包括此选项，则默认数据库将设置为 master。

(6) DEFAULT_LANGUAGE = language——指定将指派给登录名的默认语言。如果未包括此选项，则默认语言将设置为服务器的当前默认语言。即使将来服务器的默认语言发生更改，登录名的默认语言也仍保持不变。

(7) CHECK_EXPIRATION = { ON | OFF}——适用于 SQL Server 仅限的登录。指定是否应对此登录账户强制实施密码过期策略。默认值为 OFF。

(8) CHECK_POLICY = { ON | OFF}——适用于 SQL Server 仅限的登录。指定应对此登录名强制实施运行 SQL Server 的计算机的 Windows 密码策略。默认值为 ON。

(9) WINDOWS——指定将登录名映射到 Windows 登录名。

(10) CERTIFICATE certname——指定将与此登录名关联的证书名称。此证书必须已存在于 master 数据库中。

(11) ASYMMETRIC KEY asym_key_name——指定将与此登录名关联的非对称密钥的名称。此密钥必须已存在于 master 数据库中。

【例 9-1】 T-SQL 语句创建一个 SQL Server 账号 u2，密码为 123456。

```
CREATE LOGIN u2
WITH PASSWORD = '123456',
CHECK_POLICY = OFF
```

如果要通过 T-SQL 语句将前面的 WINDOWS 账号 u1 创建为 SQL Server 登录账号，则使用以下 T-SQL 命令：

```
CREATE LOGIN [YINZHIYU - PC\u1]
FROM WINDOWS
```

9.2.2 数据库用户账号

用户是数据库级的安全策略，在为数据库创建新的用户前，必须存在创建用户的一个账号或者使用已经存在的账号登录。用户登录后，如果想要操作数据库，还必须有一个数据库用户账号，然后为这个数据库用户设置某种角色，才能进行相应的操作。

1. 创建数据库用户账号

在 SSMS 中创建数据库用户的具体步骤如下：

(1) 在 SSMS 的“对象资源管理器”窗口中，展开“数据库”选项，选中要创建用户的数据库，展开此数据库，如 teaching。展开“安全性”选项，在“用户”上右击，弹出快捷菜单，从中选择“新建用户”命令，如图 9-9 所示。

(2) 在“数据库用户-新建”对话框的“常规”选项页中填写要创建的“用户名”，选择此用户的服务器“登录名”，选择“默认架构”名称，添加此用户拥有的架构，添加此用户的数据库角色，如图 9-10 所示。

(3) 在“选择页”中选择“安全对象”，进入权限设置



图 9-9 新建数据库用户

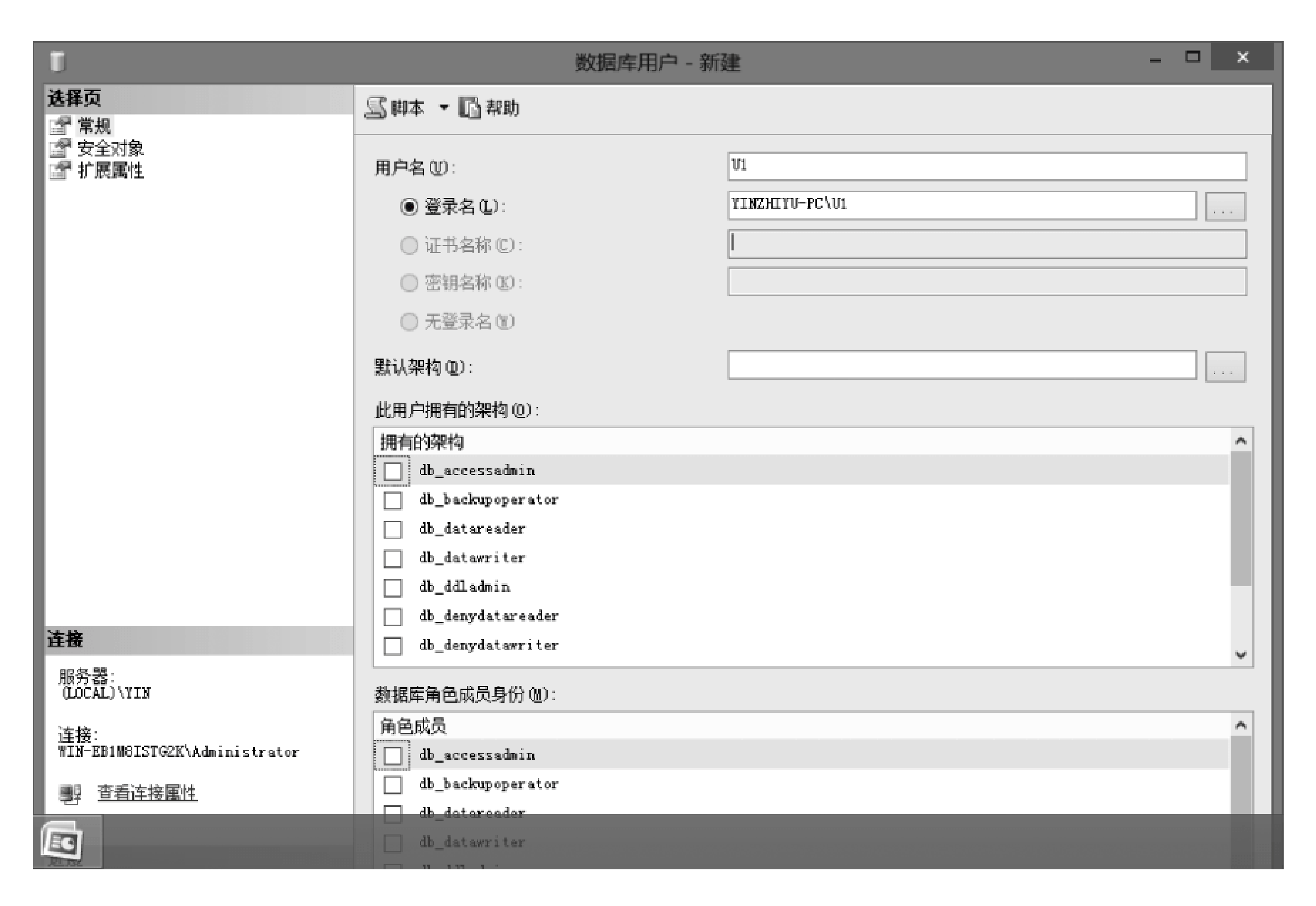


图 9-10　新建数据库用户

页面。安全对象页面主要用于设置数据库用户拥有的能够访问的数据库对象以及相应的访问权限。单击“添加”按钮为该用户添加数据库对象，并为添加的对象添加显式权限。例如，为用户添加 student 表，并设置其拥有“选择”student 表的权限，如图 9-11 所示。

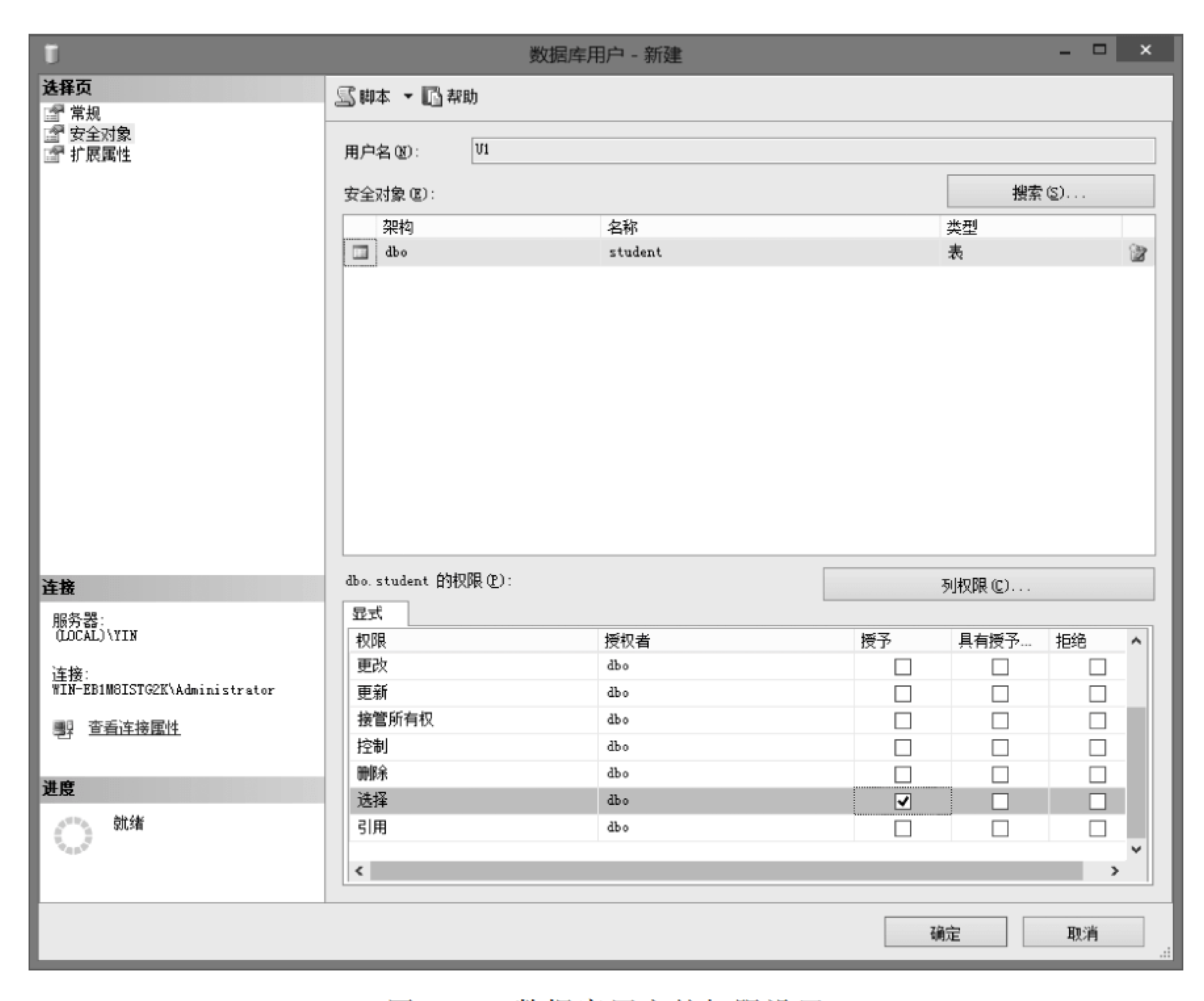


图 9-11　数据库用户的权限设置

(4) 单击“确定”按钮,完成此数据库用户的创建。

2. 查看数据库用户账号

可以使用“对象资源管理器”查看数据库的用户:在 Microsoft SQL Server Management Studio 的“对象资源管理器”面板中,展开要查看的数据库,展开“安全性”选项,再展开“用户”选项,则显示目前数据库中的所有用户,如图 9-12 所示。

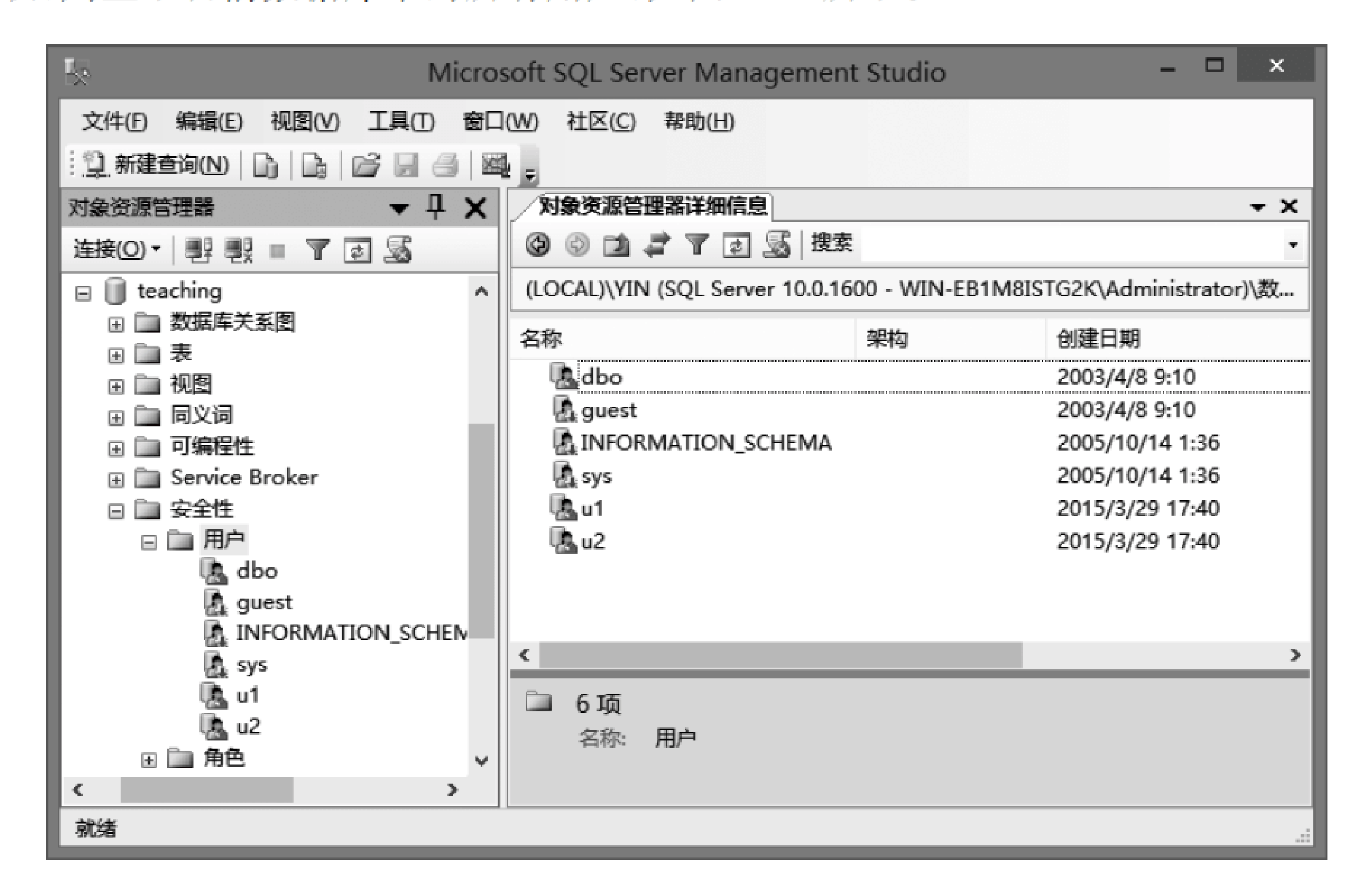


图 9-12　查看数据库用户账号

3. T-SQL 创建数据库用户

T-SQL 语句创建数据库用户的语法格式如下:

```
CREATE USER user_name
    [ { { FOR | FROM }
      {
        LOGIN login_name
        | CERTIFICATE cert_name
        | ASYMMETRIC KEY asym_key_name
      }
      | WITHOUT LOGIN
    ]
  [ WITH DEFAULT_SCHEMA = schema_name ]
```

其中,比较重要的几个参数的说明如下:

(1) user_name——指定在此数据库中用于识别该用户的名称,它的长度最多 128 个字符。在创建基于 Windows 主体的用户时,除非指定其他用户名,否则 Windows 主体名称将成为用户名。

(2) login_name——指定要为其创建数据库用户的登录名。login_name 必须是在服务器中的有效登录名。可以是基于 Windows 主体(用户或组)的登录名,也可以是使用 SQL Server 身份验证的登录名。当以此 SQL Server 登录名进入数据库时,它将获取正在创建的这个数据库用户的名称和 ID。在创建登录名从 Windows 主体映射时,使用格式

[< domainName >\< loginName >]。

(3) DEFAULT_SCHEMA = schema_name——指定服务器为此数据库用户解析对象名时将搜索的第一个架构,默认为 dbo。

【例 9-2】 用 T-SQL 语句在 teaching 数据库中创建一个数据库用户 u1,对应的服务器登录账号为 YINZHIYU-PC\u1。

```
CREATE USER u1 FOR LOGIN [YINZHIYU - PC\u1]
```

9.3 角色管理

角色是一种 SQL Server 安全账户,是 SQL Server 内部的管理单元,在管理权限时可以视为单个单元的其他安全账户的集合。角色包含 SQL Server 登录、Windows 登录、组或其他角色(与 Windows 中的用户组类似),若用户被加入到某一个角色中,则具有该角色的权限。可以建立一个角色来代表单位中一类工作人员所执行的工作,然后给这个角色授予适当的权限。

利用角色,SQL Server 管理者可以将某些用户设置为某一个角色,这样只对角色进行权限设置便可以实现对所有用户权限的设置,极大地减少了管理员的工作量。SQL Server 提供了用户通常管理工作的预定义服务器角色和数据库角色。如果有好几个用户需要在一个特定的数据库中执行一些操作,那么数据库拥有者可以在这个数据库中加入一个角色。

一般而言,角色是为特定的工作组或者任务分类而设置的,用户可以根据自己所执行的任务成为一个或多个角色的成员。当然用户可以不必是任何角色的成员,也可以为用户分配个人权限。

SQL Server 的安全体系结构中包括几个含有特定隐含权限的角色。除了两类预定义的角色,数据库拥有者还可以自己创建角色,这些角色被分成 3 类:固定服务器角色、数据库角色和应用程序角色。

9.3.1 固定服务器角色

固定服务器角色是在服务器级别定义的,所以存在于数据库外面,是属于数据库服务器的。在 SQL Server 安装时就创建了在服务器级别上应用的大量预定义的角色,每个角色对应着相应的管理权限。这些固定服务器角色用于授权给 DBA(数据库管理员),拥有某种或某些角色的 DBA 就会获得与相应角色对应的服务器管理权限。

通过给用户分配固定服务器角色,可以使用户具有执行管理任务的角色权限。根据 SQL Server 的管理任务以及这些任务相对的重要性等级来把具有 SQL Server 管理职能的用户划分为不同的用户组,每一组所具有的管理 SQL Server 的权限都是 SQL Server 内置的,即不能对其进行添加、修改和删除,只能向其中加入用户或者其他角色。因此,固定服务器角色的维护比单个权限维护更容易一些,但是固定服务器角色不能修改。

SQL Server 2008 在安装时定义了几个固定服务器角色,其具体权限描述如下:

(1) sysadmin(System Administrators)——可以在 SQL Server 中执行任何活动。

(2) serveradmin(Server Administrators)——可以设置服务器范围的配置选项,还可以

关闭服务器。

(3) setupadmin(Setup Administrators)——可以管理连接服务器和启动过程。

(4) securityadmin(Security Administrators)——可以管理登录和创建数据库的权限，还可以读取错误日志和更改密码。

(5) processadmin(Process Administrators)——可以管理在 SQL Server 中运行的进程。

(6) dbcreator(Database Creators)——可以创建、更改和删除数据库。

(7) diskadmin(Disk Administrators)——可以管理磁盘文件。

(8) bulkadmin(Bulk Administrators)——可以执行 BULK INSERT(大容量插入语句)。

在 SSMS 中，可以按以下步骤为用户分配固定服务器角色，从而使该用户获取相应的权限。

(1) 在“对象资源管理器”中，展开服务器，再展开“安全性”选项。展开“登录名”，选择要添加固定服务器角色的登录账号，如 YINZHIYU-PC\u1，右击，弹出快捷菜单，选择“属性”命令，如图 9-13 所示。

(2) 在“登录属性-YINZHIYU-PC\u1”对话框中，选择左侧“选择页”中的“服务器角色”，选择一个要为其添加的角色，比如 dbcreator，单击“确定”按钮，添加完成，如图 9-14 所示。



图 9-13　为登录账号分配固定服务器角色

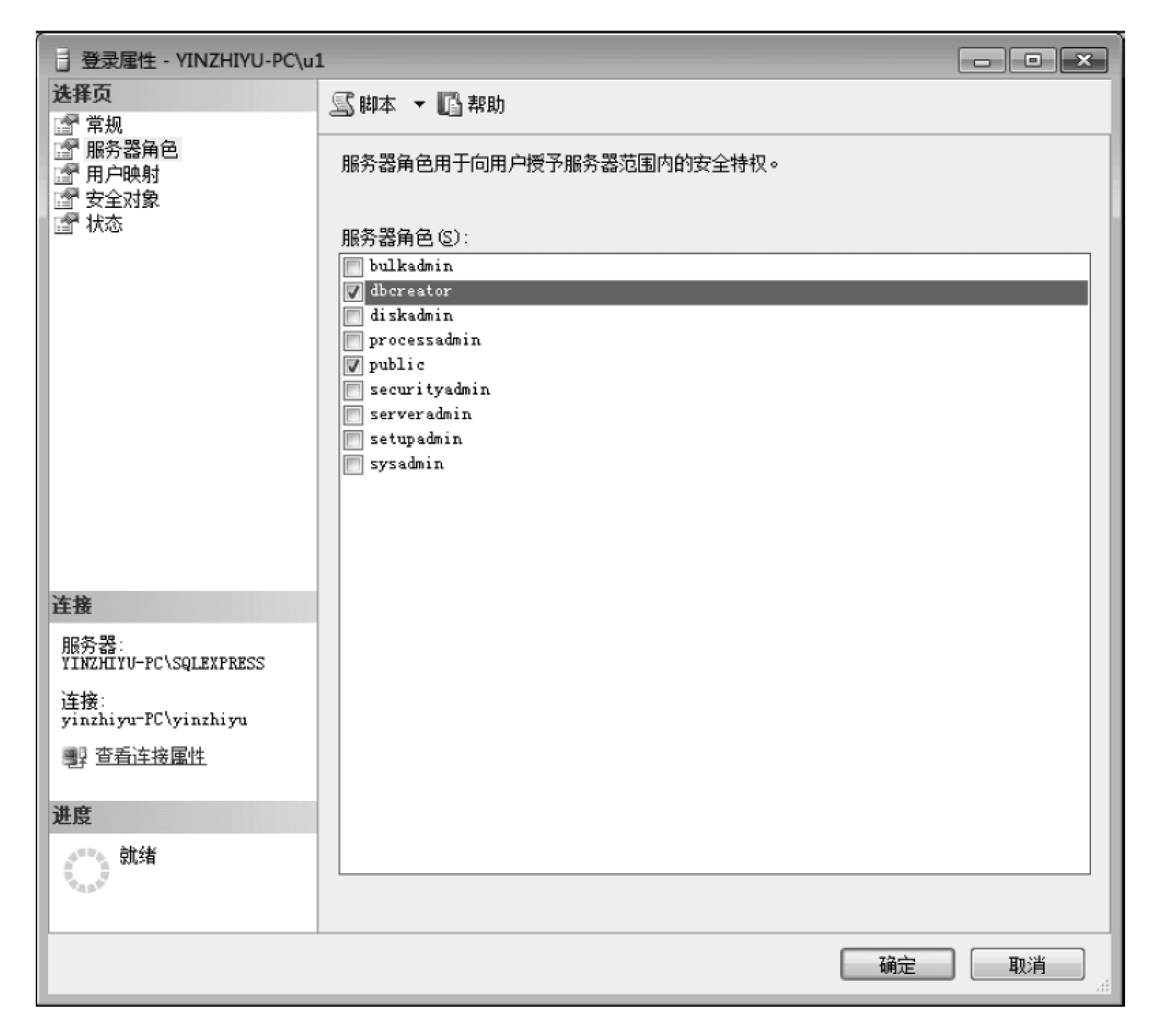


图 9-14　为登录账号添加服务器角色

9.3.2 数据库角色

在 SQL Server 2008 安装时，数据库级别上也有一些预定义的角色，在创建每个数据库时都会添加这些角色到新创建的数据库中，每个角色对应着相应的权限。这些数据库角色用于授权给数据库用户，拥有某种或某些角色的用户会获得相应角色对应的权限。

也可以为数据库添加角色，然后把角色分配给用户，使用户拥有相应的权限，在 SSMS 中，给用户添加角色(或者叫作将角色授予用户)的操作与将固定服务器角色授予用户的方法类似，通过相应角色的属性对话框可以方便地添加用户，使用户成为角色成员。

1. 固定数据库角色

固定数据库角色是为某一个用户或某一组用户授予不同级别的管理或访问数据库以及数据库对象的权限，这些权限是数据库专有的，并且还可以使一个用户具有属于同一个数据库的多个角色。

SQL Server 2008 在安装时定义了几个固定数据库角色，其具体权限描述如下：

(1) db_owner——具有数据库中的全部权限。

(2) db_accessadmin——可以添加和删除用户。

(3) db_securityadmin——可以管理全部权限、对象所有权限，拥有角色和角色成员资格。

(4) db_ddladmin——可以发出除 GRANT、REVOKE、DENY 之外的所有数据定义语句。

(5) db_backupoperator——具有备份数据库的权限。

(6) db_datareader——可以选择数据库内任何用户表中的所有数据。

(7) db_datawriter——可以更改数据库内任何表中的所有数据。

(8) db_denydatareader——不能选择数据库内任何用户表中的任何数据。

(9) db_denydatawriter——不能更改数据库内任何用户表中的任何数据。

(10) public——最基本的数据库角色。每个用户可以不属于其他 9 个固定数据库的角色，但至少会属于 public 数据库角色，当在数据库中添加新用户账号时，SQL Server 2008 会自动将新的用户账号加入 public 数据库角色中。

2. 自定义数据库角色

创建用户自定义的数据库角色就是创建一组用户，这些用户具有相同的一组权限。如果一组用户需要执行在 SQL Server 中指定的一组操作并且不存在对应的 Windows 组，或者没有管理 Windows 用户账号的权限，就可以在数据库中建立一个用户自定义的数据库角色。

另外，创建用户自定义数据库角色时，创建者需要完成下列任务：

① 创建新的数据库角色。

② 分配权限给创建的角色。

③ 将这个角色授予某个用户。

在 SSMS 中创建用户自定义的数据库角色操作的具体步骤如下：

(1) 在 SSMS 的“对象资源管理器”中，展开要添加新角色的目标数据库，展开“安全性”选项，再展开“角色”选项，右击“数据库角色”选项，在弹出的快捷菜单中选择“新建数据库角

色”命令，如图 9-15 所示。



图 9-15　新建数据库角色

(2) 在“数据库角色-新建”对话框的“常规”选项页中，添加“角色名称”和“所有者”，并选择此角色所拥有的架构。在此对话框中也可以单击“添加”按钮为新创建的角色添加用户，如图 9-16 所示。



图 9-16　“数据库角色-新建”对话框

(3) 选择“选择页”中的“安全对象”选项，单击“搜索”按钮，出现“添加对象”对话框，如图 9-17 所示。

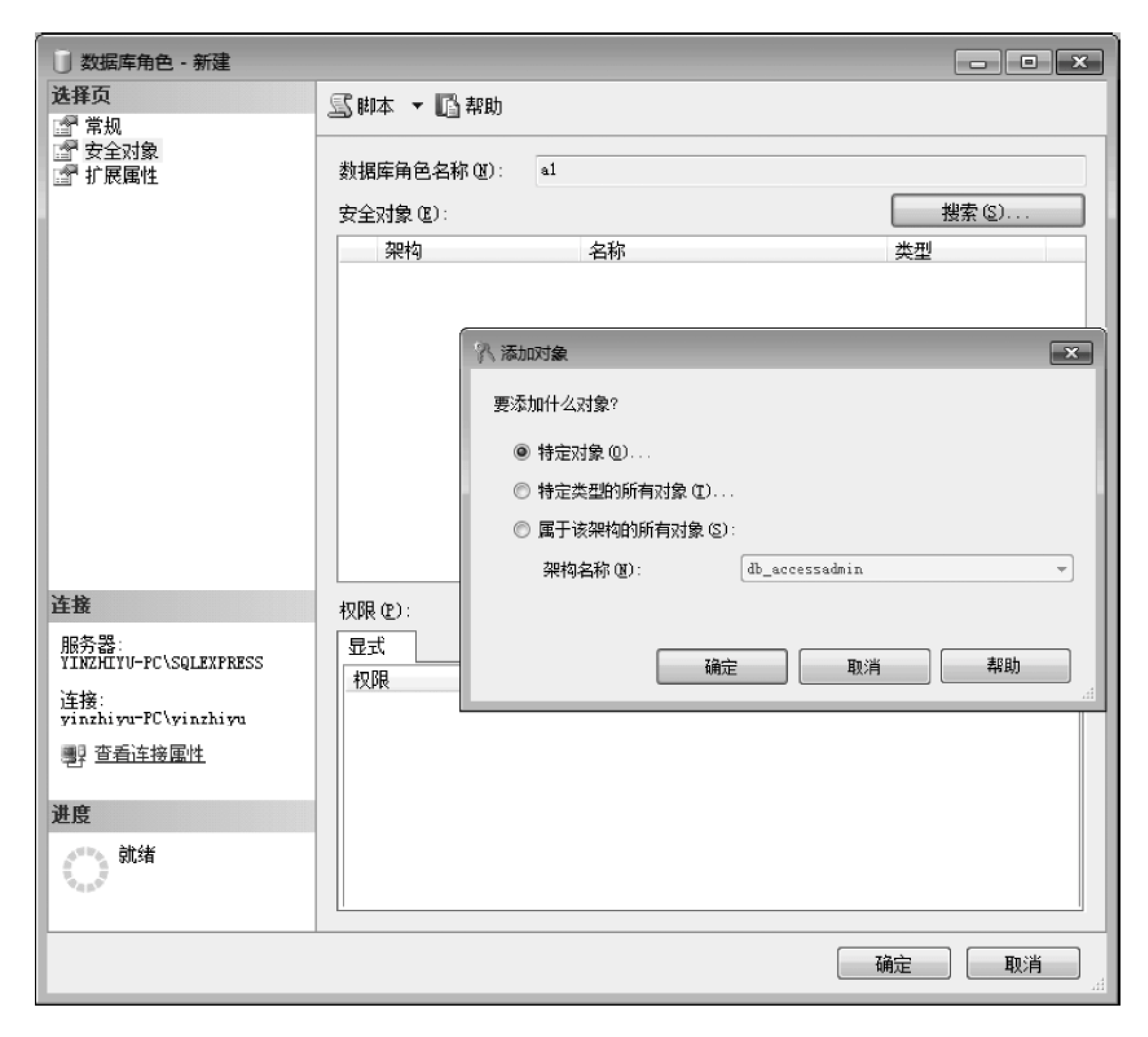


图 9-17 “添加对象”对话框

(4) 选中“特定对象”单选按钮,单击“确定”按钮,出现“选择对象”对话框,单击“对象类型”按钮,出现“选择对象类型”对话框,这里选择“表”选项,单击“确定”按钮,如图 9-18 所示。

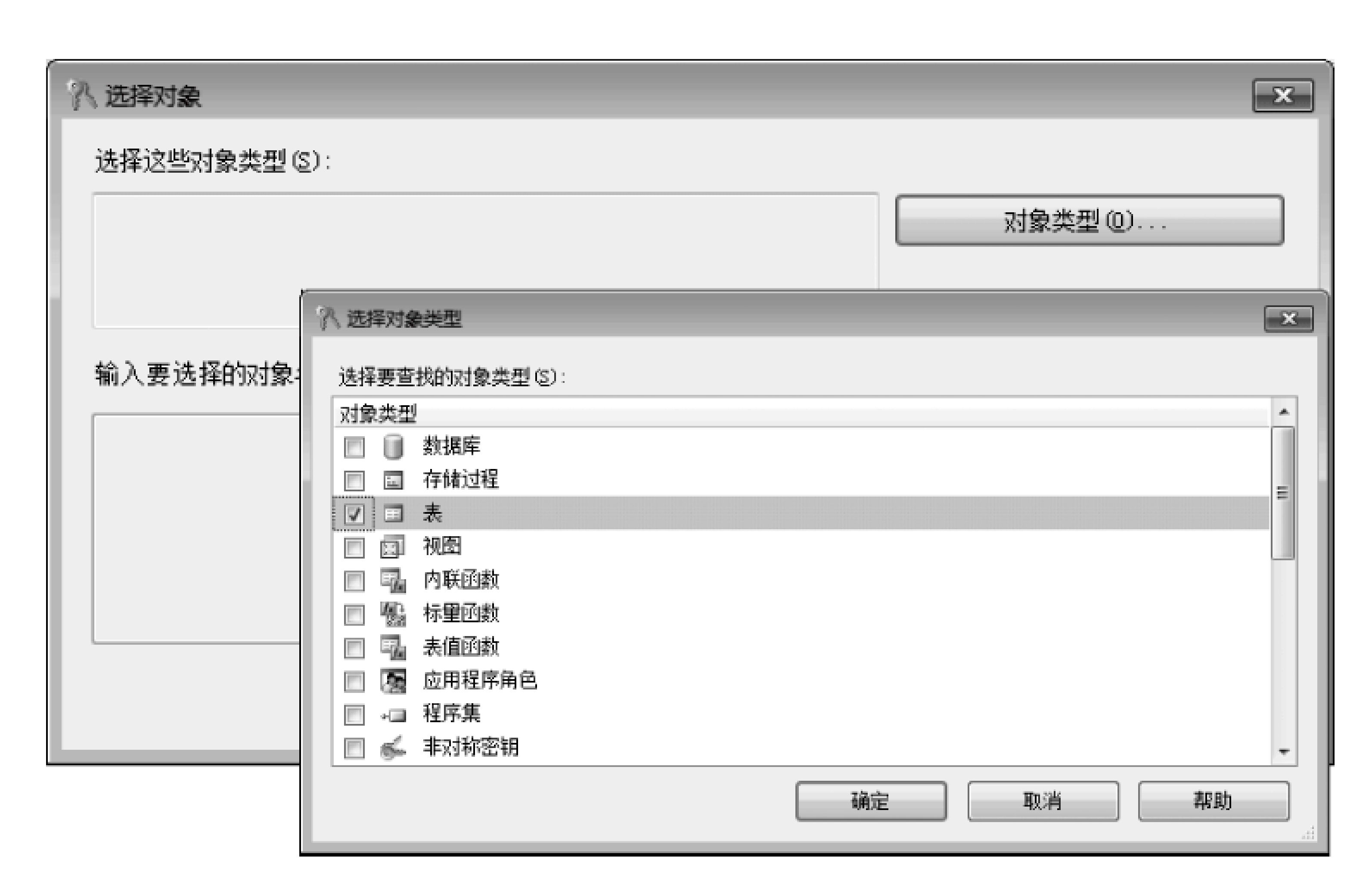


图 9-18 选择对象类型

(5) 回到“选择对象”对话框,单击“浏览”按钮,出现“查找对象”对话框,选择设置此角色的表,如 student 表,如图 9-19 所示。



图 9-19 “选择对象”—“表”

(6) 单击“确定”按钮。接着进行权限设置,可以为新创建的角色添加所拥有的数据库对象的访问权限,如 student 表的数据“更新”和“选择”权限,如图 9-20 所示。

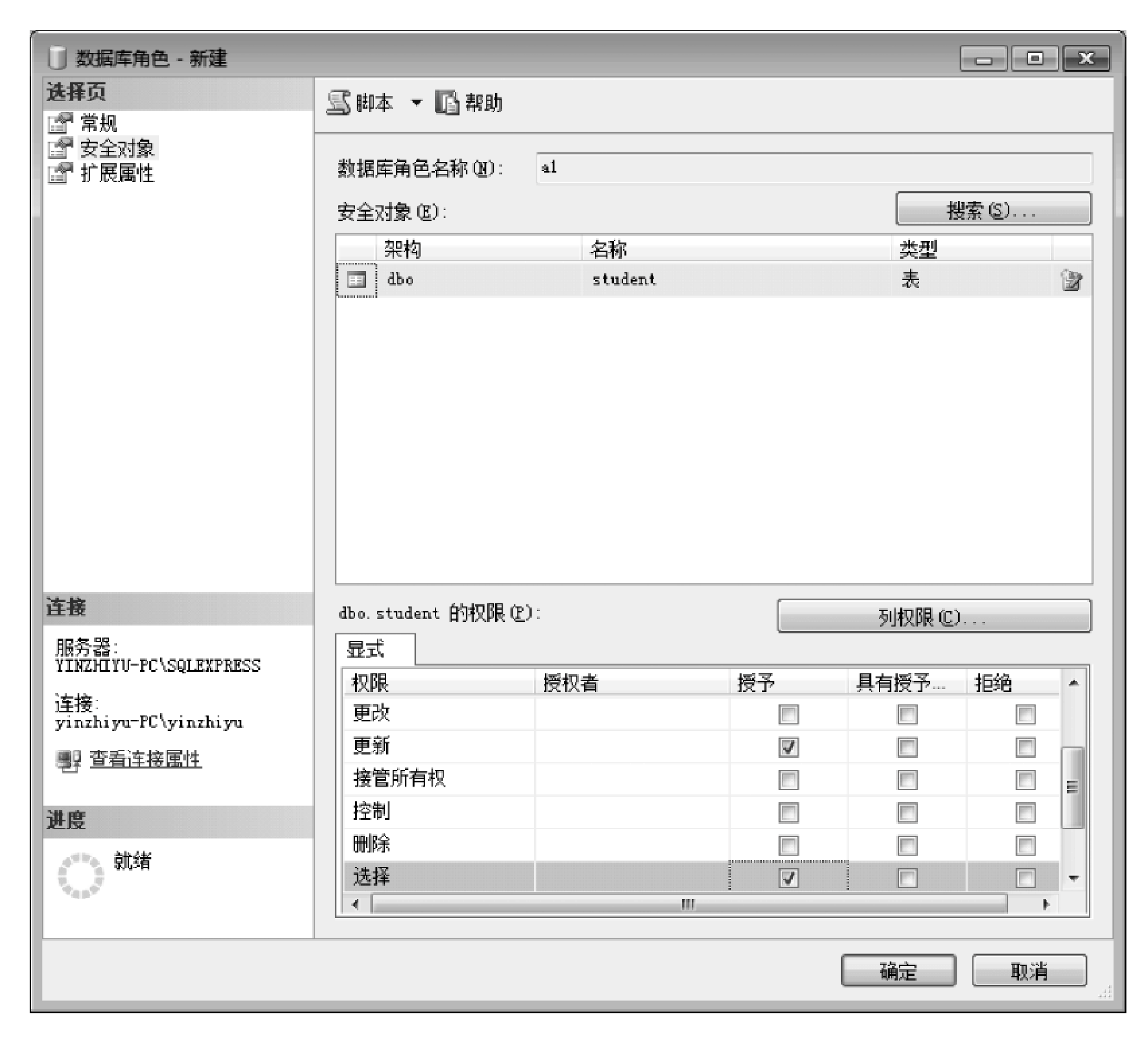


图 9-20 为新创建的角色添加数据库对象的访问权限

（7）单击“确定”按钮，自定义数据库角色创建完成。

9.3.3 应用程序角色

应用程序角色是一种比较特殊的由用户定义的数据库角色。

应用程序角色是用来控制应用程序存取数据库的，本身不包含任何成员。在编写数据库的应用程序时，可以自定义应用程序角色，让应用程序的操作者能用编写的程序来存取SQL Server的数据，也就是说，应用程序的操作者本身并不需要在SQL Server 2008上拥有登录账号以及用户账号，但是仍然可以存取数据库。

如果想让某些用户只能通过特定的应用程序间接地存取数据库中的数据而不是直接地存取数据库数据时，就应该考虑使用应用程序角色。当某一个用户使用了应用程序角色时，便放弃了已被赋予的所有数据库专有权限，所拥有的只是应用程序角色被设置的权限。

应用程序角色可以加强对某一个特别的应用程序的安全性。换句话说，允许应用程序自己代替SQL Server接管用户验证的职责。例如，某公司职员只是用某个特定的应用程序来修改员工数据信息，那么就可以建立应用程序角色。

应用程序角色和所有其他的角色都有很大不同，主要表现在以下两个方面：

（1）应用程序角色没有成员，因为它们只在应用程序中使用，不需要直接对某些用户赋予权限。

（2）必须为应用程序角色设计一个密码以激活它。

当应用程序角色被应用程序的会话激活以后，会话就会失去所有属于登录、用户账号或角色的权限，因为这些角色都只适用于它们所在的数据库内部，所以会话只能通过guest用户账号的权限来访问其他数据库。因此，如果在数据库中没有guest用户账号，会话就不能获得访问数据库的权限。

在SSMS中创建应用程序角色的步骤如下：

（1）在SSMS的“对象资源管理器”中展开要建立应用程序角色的目标数据库，展开“安全性”选项，再展开“角色”选项，在“应用程序角色”选项上右击，弹出快捷菜单，选择“新建应用程序角色”命令。

（2）在“应用程序角色-新建”对话框中，如图9-21所示，输入角色名称、密码等。

（3）下面的步骤与创建“数据库角色”相同。

当一个数据库连接启动以后，必须执行系统存储过程sp_setapprole来激活应用程序角色所拥有的权限。这个过程的语法格式为：

```
sp_setapprole
[@rolename] 'role'[@passwd = ] 'password'[,[@encrypt = ] 'encrypt_style']
```

其中，role是当前数据库中已经定义过的应用程序角色的名称；password表示密码；encrypt_style定义密码的加密模式。

注意：激活应用程序角色以后，就不能让这个角色无效，而必须等到会话断开之后；应用程序角色总是和数据库绑定的，也就是说，它作用的范围是当前数据库。如果在会话中改变了当前数据库，那么只能执行那个数据库中允许的操作。

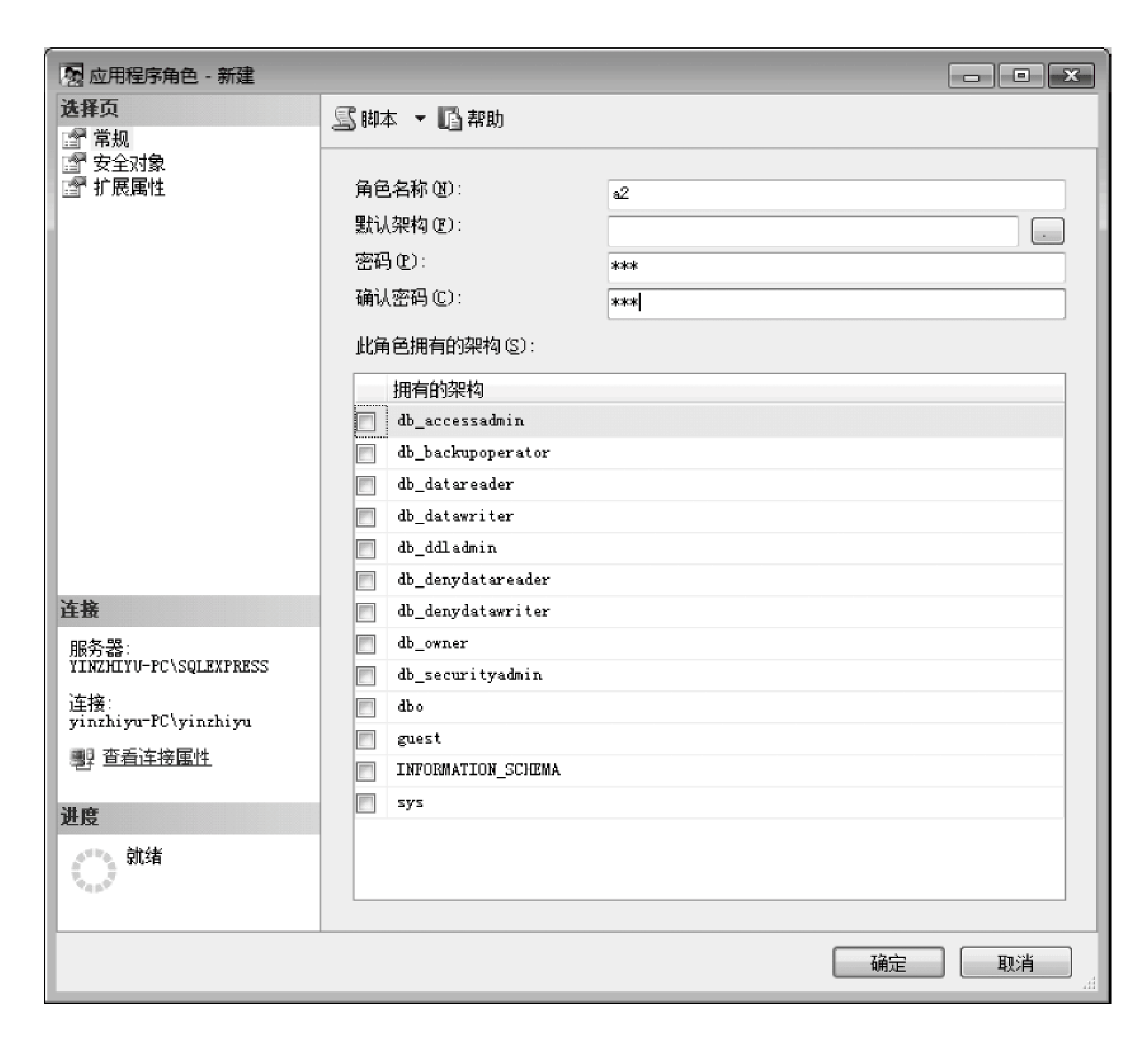


图 9-21 创建应用程序角色

9.4 权限管理

权限用于控制对数据库对象的访问,以及指定用户对数据库可以执行的操作,用户在登录到 SQL Server 之后,其用户账号所归属的 Windows 组或角色所被赋予的权限决定了该用户能够对哪些数据库对象执行哪种操作以及能够访问、修改哪些数据。

9.4.1 权限的类别

用户可以设置服务器和数据库的权限。服务器权限允许数据库管理员执行管理任务,数据库权限用于控制对数据库对象的访问和语句执行。用户只有在具有访问数据库的权限之后,才能够对服务器上的数据库进行权限下的各种操作。

1. 服务器权限

服务器权限允许数据库管理员执行任务。这些权限定义在固定服务器角色中。这些固定服务器角色可以分配给登录用户,但这些角色是不能修改的。一般只把服务器权限授予 DBA(数据库管理员),他不需要修改或者授权给其他用户登录。在 9.3.1 节中已有过详细介绍,此处不再赘述。

2. 数据库对象权限

数据库对象是授予用户允许他们访问数据库中对象的一类权限,对象权限对于使用

SQL 语句访问表或者视图是必需的。除了数据库中的对象权限外，还可以给用户分配数据库权限。SQL Server 2008 对数据库权限进行了扩充，增加了许多新的权限，这些数据库权限除了授权用户可以创建数据库对象和进行数据库备份外，还增加了一些更改数据库对象的权限。

9.4.2 权限操作

SQL Server 2008 中的权限控制操作可以通过在 SSMS 中对用户的权限进行设置，也可以使用 T-SQL 提供的 GRANT(授予)、REVOKE(撤销)和 DENY(禁止)语句完成。

1. 在 SSMS 中设置权限

在 SSMS 中给用户设置权限的具体步骤如下：

(1) 在 SSMS 的"对象资源管理器"中展开目标数据库的"用户"选项，如图 9-22 所示，在目标用户上右击，在快捷菜单中选择"属性"命令。

(2) 在"数据库用户"对话框中选择"选择页"窗口中的"安全对象"选项，进入权限设置页面，单击"搜索"按钮，在"添加对象"对话框中，单击要添加的对象类别前的单选按钮(如"特定对象")，添加权限的对象类别，然后单击"确定"按钮，如图 9-23 所示。

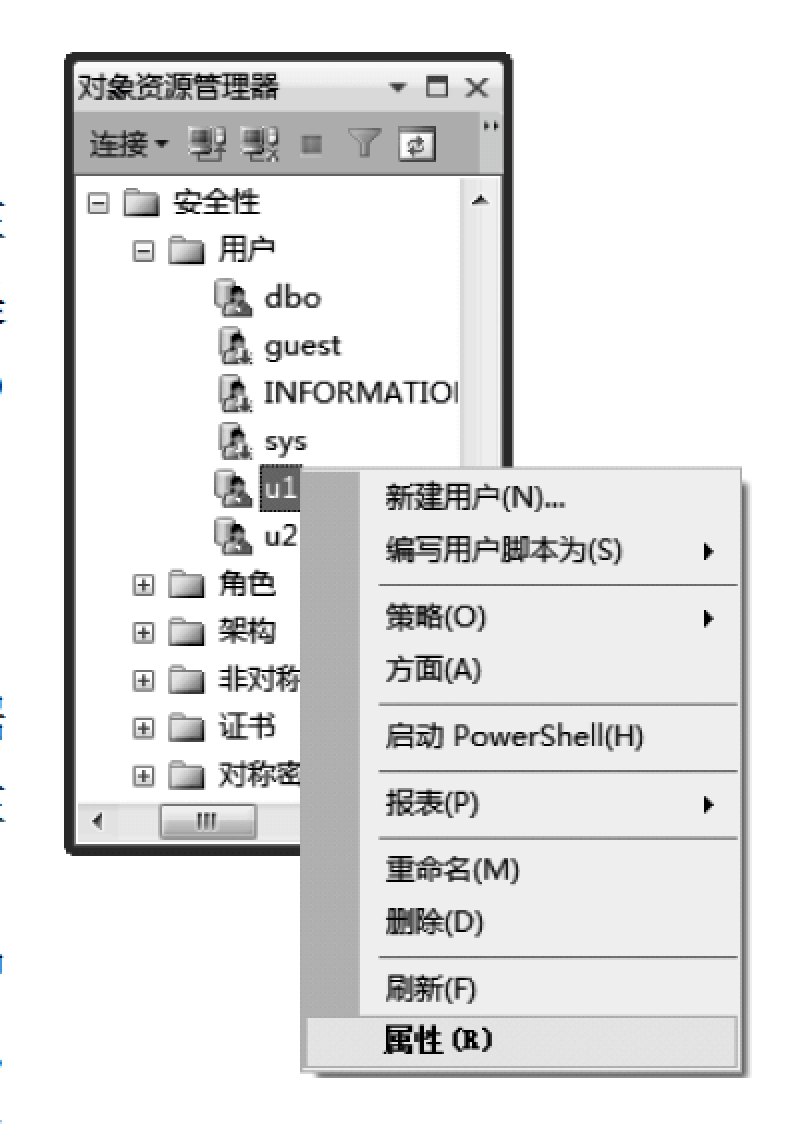


图 9-22 在对象资源管理器中为用户添加对象权限

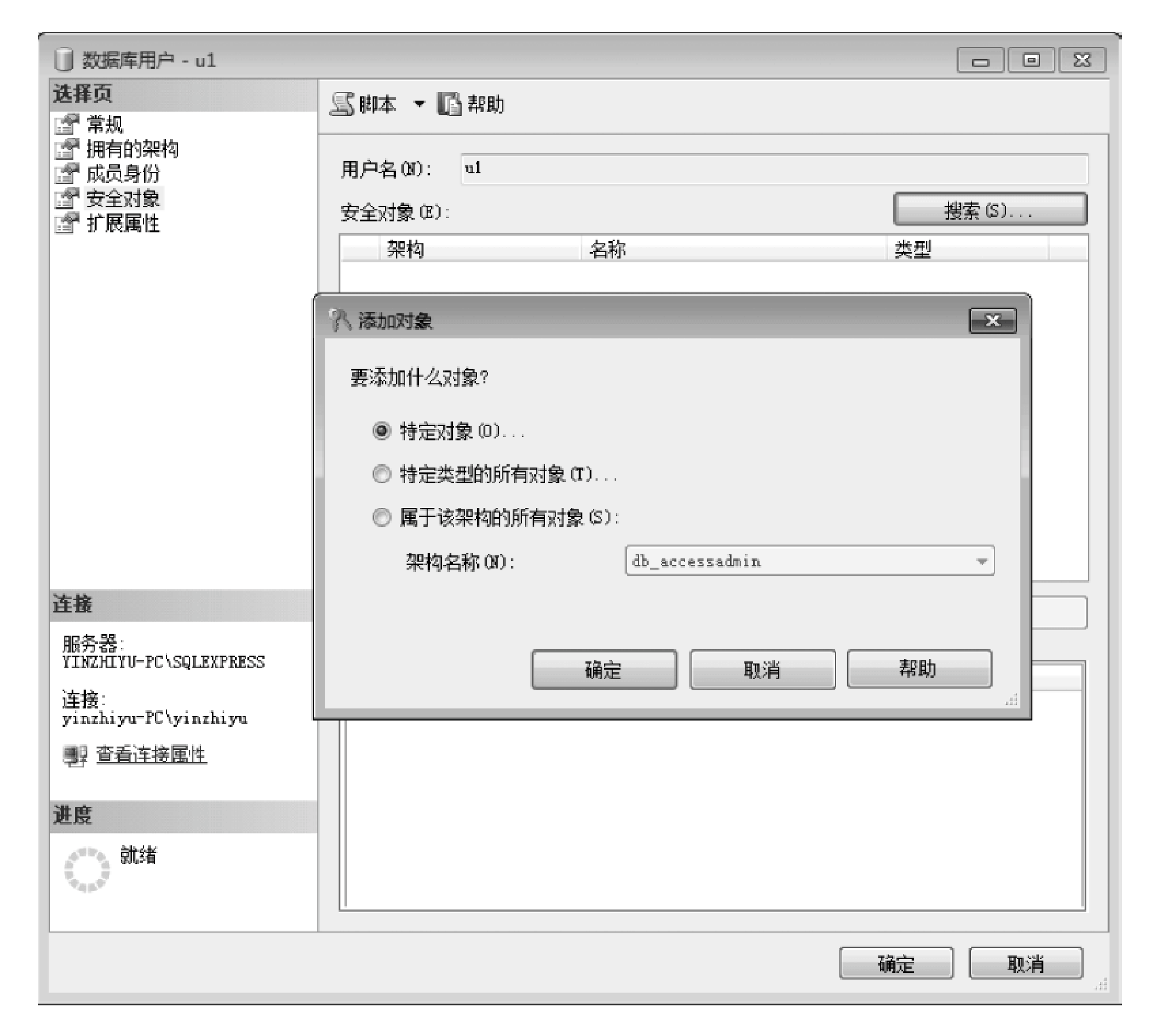


图 9-23 "数据库用户"及"添加对象"对话框

(3) 在“选择对象”对话框中，单击“对象类型”按钮，进入“选择对象类型”对话框，选择需要添加权限的对象类型前的复选框，如图 9-24 所示，最后单击“确定”按钮。



图 9-24 “选择对象”及“选择对象类型”对话框

(4) 回到“选择对象”对话框，在该对话框中出现了刚才选择的对象类型，单击该对话框中的“浏览”按钮。在“查找对象”对话框中，依次选中要添加权限的对象前的复选框，单击“确定”按钮，如图 9-25 所示。

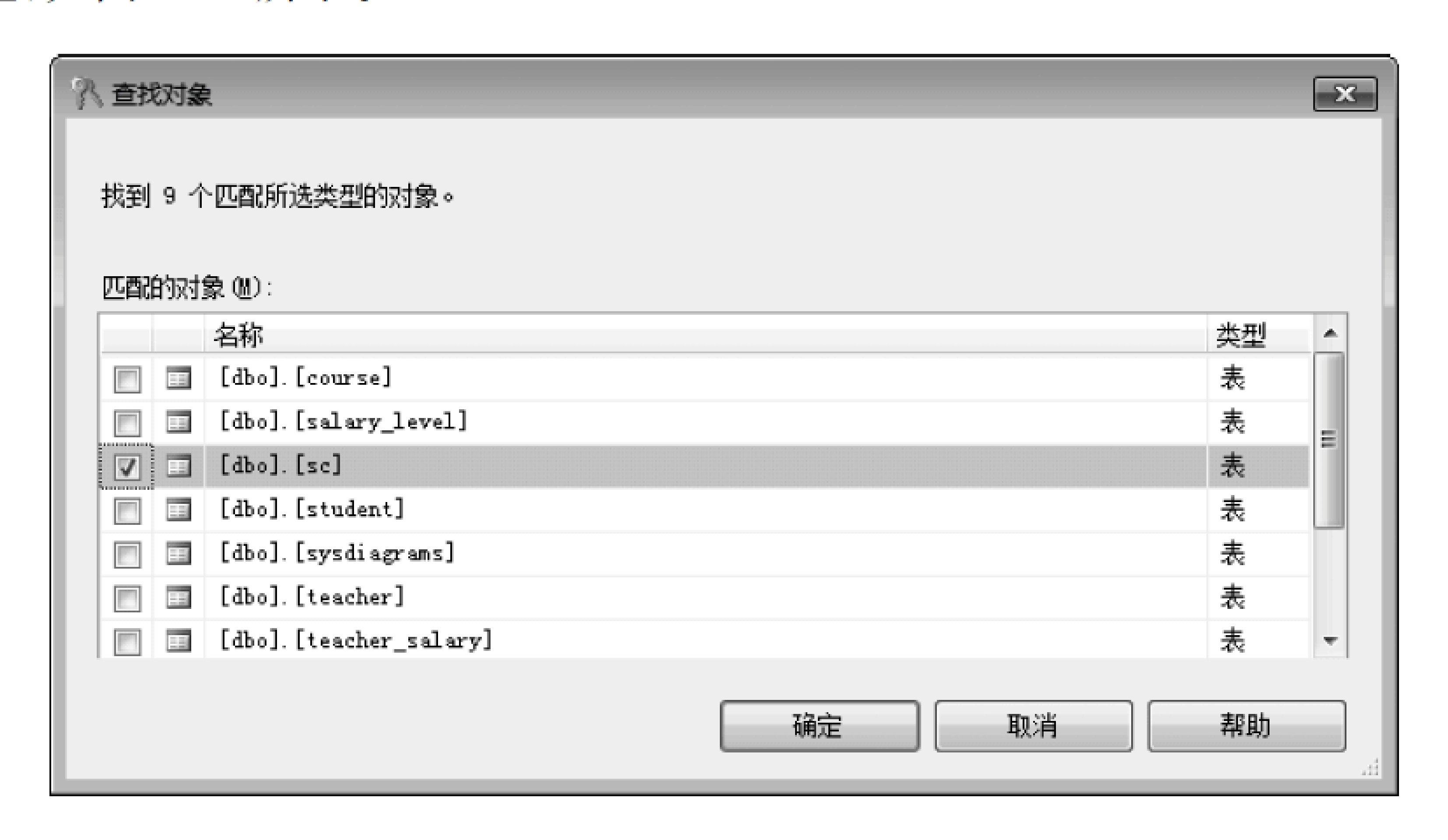


图 9-25 “查找对象”对话框

(5) 再次回到“选择对象”对话框，其中已包含了选择的对象，确定无误后，单击该对话框中的“确定”按钮，完成对象选择操作。回到“数据库用户”对话框，其中已包含用户添加的对象，依次选择每一个对象，并在下面的该对象的“显式权限”窗口中根据需要选择“授予和拒绝”列的复选框，添加或禁止对该(表)对象的相应访问权限。设置完每一个对象的访问权限后，单击“确定”按钮，完成给用户添加数据库对象权限所有操作，如图 9-26 所示。

2. 使用 T-SQL 设置权限

数据库内的权限始终授予数据库用户、角色和 Windows 用户或组，但从不授予 SQL



图 9-26　为数据库用户设置权限

Server 登录。为数据库内的用户或角色设置适当权限的方法有：GRANT 授予权限、DENY 禁止权限和 REVOKE 撤销权限。

1）授权语句

T-SQL 语句中的 GRANT 命令的语法格式如下：

```
GRANT { ALL [ PRIVILEGES ] }
      | permission [ ( column [ ,...n ] ) ] [ ,...n ]
      [ ON [ class :: ] securable ] TO principal [ ,...n ]
      [ WITH GRANT OPTION ] [ AS principal ]
```

其中的参数详细说明如下：

(1) ALL——不推荐使用此选项，保留此选项仅用于向后兼容。它不会授予所有可能的权限。授予 ALL 参数相当于授予以下权限：

- 如果安全对象是数据库，则 ALL 对应 BACKUP DATABASE、BACKUP LOG、CREATE DATABASE、CREATE DEFAULT、CREATE FUNCTION、CREATE PROCEDURE、CREATE RULE、CREATE TABLE 和 CREATE VIEW。
- 如果安全对象是标量函数，则 ALL 对应 EXECUTE 和 REFERENCES。
- 如果安全对象是表值函数，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

- 如果安全对象是存储过程，则 ALL 表示 EXECUTE。
- 如果安全对象是表，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。
- 如果安全对象是视图，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

(2) PRIVILEGES——包含此参数是为了符合 ISO 标准。

(3) permission——权限的名称。

(4) column——指定表中将授予其权限的列的名称。需要使用括号( )。

(5) class——指定将授予其权限的安全对象的类。需要使用范围限定符::。

(6) securable——指定将授予其权限的安全对象。

(7) TO principal——主体的名称。可为其授予安全对象权限的主体，随安全对象的不同而不同。

(8) GRANT OPTION——指示被授权者在获得指定权限的同时还可以将指定权限授予其他主体。

(9) AS principal——指定一个主体，执行该查询的主体从该主体获得授予该权限的权利。

在以下例子中，首先假定所有被授权的登录用户已存在。

【例 9-3】 把查询学生表 student 的权限授予用户 u1。

```
GRANT SELECTON student TO u1
```

执行此操作后，用户 u1 就被授予了查询 student 表的权限。可以在 SSMS 中查看到用户 u1 被授予了 student 表的 SELECT 权限。展开 teaching 数据库的"用户"选项，在目标用户 u1 上右击，在快捷菜单中选择"属性"命令，如图 9-27 所示。



图 9-27 查看用户 u1 的属性

在"数据库用户"对话框中选择"选择页"窗口中的"安全对象"选项，可以看到 u1 拥有了对 sc 表的"选择"权限，并被拒绝了 sc 表的"数据更新"权限，它不能通过用户拥有的角色或 WINDOWS 组继承 sc 表的"数据更新"权限，如图 9-28 所示。此时，u1 登录 SQL Server 就可以对 sc 表进行 SELECT 操作。

【例 9-4】 把 student 表的所有数据操作权限授予用户 u2 和 u3。

```
GRANT SELECT,INSERT,UPDATE,DELETE ON student TO u2,u3
```

执行此操作后，用户 u2 和 u3 就被授予了 student 表的 SELECT、INSERT、UPDATE、DELETE 权限。可以在 SSMS 中查看到用户 u2 和 u3 被授予了 student 表的 SELECT、INSERT、UPDATE、DELETE 权限。此时，u2 或 u3 登录 SQL Server 就可以对 student 表进行所有这些操作。



图 9-28　用户 u1 的“选择”权限

【例 9-5】　把对选课表 SC 的查询权限授予所有用户。

```
GRANT SELECT ON SC TO PUBLIC
```

执行此操作后，所有用户都被授予了 SC 表的查询权限。可以在 SSMS 中查看到 PUBLIC 用户被授予了 SC 表的 SELECT 权限。此时，所有登录用户都可以对 SC 表进行 SELECT 操作。

【例 9-6】　把查询 student 表和修改学生学号的权限授予用户 u4。

```
GRANT SELECT,UPDATE(sno) ON student TO u4
```

【例 9-7】　把对 SC 表插入数据的权限授予用户 u5，并允许 u5 再将此权限授予其他用户。

```
GRANT INSERT ON SC TO u5 WITH GRANT OPTION
```

执行此操作后，用户 u5 被授予了 SC 表插入数据的权限，同时允许 u5 再将此权限授予其他用户。例如，下面就是用户 u5 为用户 u6 授予了 SC 表插入数据的权限，并允许 u6 再将此权限授予其他用户。

```
GRANT INSERT ON SC TO u6 WITH GRANT OPTION
```

执行此操作后，用户 u6 被授予了 SC 表插入数据的权限，同时允许 u6 再将此权限授予其他用户。例如，下面就是用户 u6 为用户 u7 授予了 SC 表插入数据的权限，u7 就不能再将

此权限授予其他用户了。

```
GRANT INSERT ON SC TO u7
```

【例 9-8】 将查询表值函数 st_func 权限授予用户 u2。

```
GRANT SELECT ON st_func TO u2
```

【例 9-9】 将 teaching 数据库中建表的权限授予 u8。

```
USE teaching
GRANT create table TO u8
```

2）撤销权限语句

T-SQL 语句中的 REVOKE 命令的语法格式如下：

```
REVOKE [ GRANT OPTION FOR ]
      {
        [ ALL [ PRIVILEGES ] ]
        |permission [ ( column [ ,…n ] ) ] [ ,…n ]
      }
       [ ON [ class :: ] securable ]
       { FROM } principal [ ,…n ]
       [ CASCADE] [ AS principal ]
```

其中，与 GRANT 命令中不同的参数说明如下：

（1）GRANT OPTION FOR——指示将撤销授予指定权限的能力。在使用 CASCADE 参数时，需要具备该功能。

（2）CASCADE——指示当前正在撤销的权限也将从其他被该主体授权的主体中撤销。使用 CASCADE 参数时，还必须同时指定 GRANT OPTION FOR 参数。

【例 9-10】 把用户 u4 修改 student 表学号的权限撤销。

```
REVOKE UPDATE(sno) ON student FROM u4
```

【例 9-11】 撤销所有用户对 SC 表的查询权限。

```
REVOKE SELECT ON SC FROM PUBLIC
```

【例 9-12】 把用户 u5 对 SC 表的 INSERT 权限撤销。

```
REVOKE INSERT ON SC FROM u5 CASCADE
```

注意：执行此操作后，用户 u5 被撤销了 SC 表插入数据的权限，u5 授予其他用户的此权限也被一并撤销，包括 u6 和 u7。

3）禁止权限语句

T-SQL 语句中的 DENY 命令的语法格式如下：

```
DENY { ALL [ PRIVILEGES ] }
       | permission [ ( column [ ,…n ] ) ] [ ,…n ]
       [ ON [ class :: ] securable ] TO principal [ ,…n ]
       [ CASCADE] [ AS principal ]
```

其中参数的含义与 GRANT 和 REVOKE 命令完全相同。

DENY 语句拒绝对 SQL Server 2008 的特定数据库对象的权限，防止主体通过其组或角色成员身份继承权限。

【例 9-13】 拒绝用户 u1 对课程表的 SELECT 权限。

```
DENY SELECT ON course TO u1
```

【例 9-14】 拒绝 u2 对存储过程 GetStudent 的 EXECUTE 权限。

```
DENY EXECUTE ON GetStudent TO u2
```

习　题　9

1. 简答题

(1) 简述服务器登录账户和数据库用户账号的创建方法。

(2) SQL Server 两种身份验证的优点分别是什么？

(3) 简述角色的概念及其分类。

(4) 什么是固定服务器角色？什么是固定数据库角色？

(5) 试述用户自定义数据库角色及应用程序角色的建立方法。

(6) 简述进行权限设置时，授予、拒绝或撤销三者的关系。

2. 操作题

(1) 创建 windows 账号 user1，并将 user1 创建为 teaching 的用户。

界面方式完成以下题目：

赋予 user1 用户 db_datareader 角色；

授予 user1 用户更新 student 表中数据的权限；

user1 登录 SQL Server，对数据库中所有表进行查询操作；对 student 表和 course 表的某些列进行修改操作；并分析为什么有些操作不能执行。

(2) 创建一个 SQL Server 账号 user2，并将 user2 创建为 bankcard 数据库的用户。

T-SQL 语句完成以下题目：

授予 user2 用户查询 Trecord 表和 depositor 表的权限；

授予 user2 用户修改 depositor 表中 VIP 列数据的权限；

user2 登录 SQL Server，对 Trecord 表和 depositor 表进行查询操作；对 depositor 表的 VIP 列数据进行修改操作，再对 account 表的 Balance 列进行修改操作；并分析为什么有些操作不能执行。

第 10 章 数据加密和攻击防范

加密是指通过使用加密算法和密钥对数据进行模糊处理的过程。在 SQL Server 中，加密并不能替代其他的安全设置，比如防止未被授权的人访问数据库或是数据库实例所在的 Windows 系统，而是作为当数据库被破解或是备份被窃取后的最后一道防线。通过加密，使得未被授权的人在没有密钥或解密算法的情况下所窃取的数据变得毫无意义。

黑客对 SQL Server 数据库的各种攻击是相对危险的数据库安全威胁，所以攻击防范也是数据库安全保护的重要环节。

本章主要讲述 SQL Server 中数据加密的几种方法以及黑客攻击和防范措施。

10.1 数据加密

加密是一种帮助保护数据的机制，它通过特定的加密(encryption)算法和密钥(secret key)将数据变为乱码，使原始数据转为不可读形式，只有经过授权的访问者才能使用解密(decryption)算法和密钥将数据解密，以实现正确的读取。

10.1.1 数据加密简介

数据加密和解密的最简单过程如图 10-1 所示。

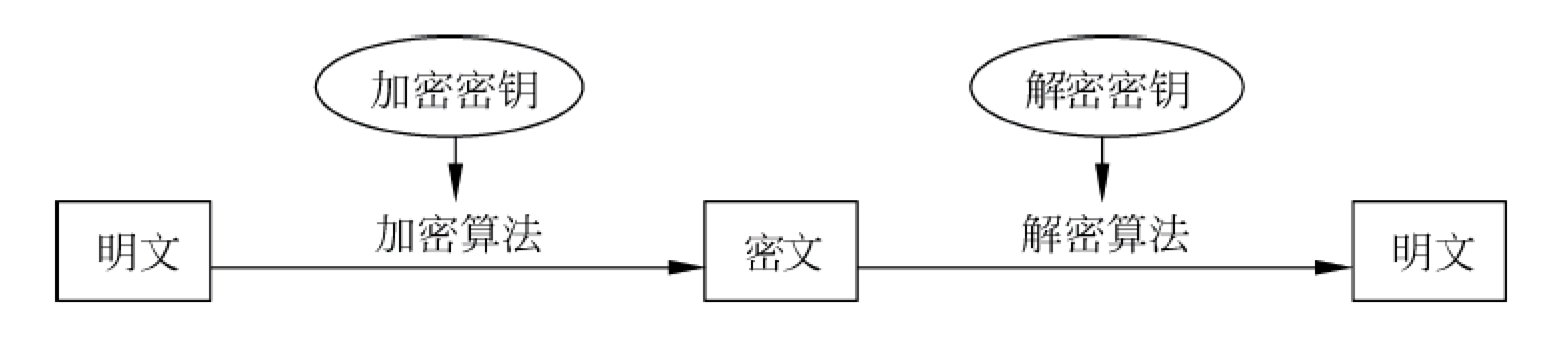


图 10-1 加密解密的一般过程

1. 数据加密分类

一般来说，加密可以分为两大类：对称加密(Symmetric Cryptography)和非对称加密(Asymmetric Cryptography)。

对称加密是最快速、最简单的一种加密方式，加密与解密用的是同样的密钥。对称加密有很多种算法，由于它效率很高，所以被广泛用于很多加密协议的核心中。

对称加密通常使用的是相对较小的密钥，一般小于 256bit。因为密钥越大，加密性越强，但加密与解密的过程越慢。如果你只用 1bit 来做这个密钥，那黑客们可以先试着用 0 来解密，不行的话就再用 1 解；但如果你的密钥有 1MB 大，黑客们可能永远也无法破解，但加密和解密的过程要花费很长的时间。所以，密钥的大小既要照顾到安全性，也要照顾到效率。

非对称加密为数据的加密与解密提供了一个非常安全的方法，它使用了一对密钥：公钥(public key)和私钥(private key)。私钥只能由一方安全保管，不能外泄，而公钥则可以发给任何请求它的人。非对称加密使用这对密钥中的一个进行加密，而解密则需要另一个密钥。比如，你向 A 请求他的公钥，A 将公钥发送给你；你使用公钥对消息加密，那么只有持有对应私钥的人(A)才能对你的消息解密。

对于对称加密而言，因为使用数据时不仅仅需要传输数据本身，还要通过某种方式传输密钥，这很有可能使得密钥在传输的过程中被窃取。而对于非对称加密，私钥持有人可以把公钥发送给任何人，但不需要将私钥发送出去，因此安全性大大提高。

对称加密算法简单、效率高，非对称加密算法复杂、效率低，因此，一种折中的办法是使用对称密钥来加密数据(数据一般较大)，而使用非对称密钥来加密对称密钥(密钥一般较小)。这样既可以利用对称密钥的高性能，也可以利用非对称密钥的可靠性。

SQL Server 中还有第三类加密方式：证书。使用证书是非对称加密的另一种形式，但一个组织可以使用证书并通过数字签名将一对公钥和私钥与其拥有者相关联，从而提高可靠性。

2. SQL Server 中的加密简介

SQL Server 2000 和以前的版本是不支持加密的。所有的加密操作都需要在程序中完成。这导致了一个问题：数据库中加密的数据仅仅是对某一特定程序有意义，而另外的程序如果没有对应的解密算法，则数据变得毫无意义。

SQL Server 2005 引入了列级加密。可以对数据库表中特定列进行加密，只保护“用户名”“密码”“银行卡号”等敏感数据，既有利于数据的安全性，又可以提高数据库的访问速度。这个过程涉及 4 对加密和解密的内置函数，包括：

(1) EncryptByCert()和 DecryptByCert()——利用证书对数据进行加密和解密。

(2) EncryptByAsymKey()和 DecryptByAsymKey()——利用非对称密钥对数据进行加密和解密。

(3) EncryptByKey()和 DecryptByKey()——利用对称密钥对数据进行加密和解密。

(4) EncryptByPassphrase()和 DecryptByPassphrase()——利用密码字段产生对称密钥对数据进行加密和解密。

SQL Server 2008 引入了透明数据加密(Transparent Data Encryption，TDE)。所谓透明数据加密，就是加密在数据库中进行，但从程序的角度来看就好像没有加密一样。和列级加密不同的是，TDE 加密的级别是整个数据库。使用 TDE 加密的数据库文件不能附加到另一个没有证书的实例上；而使用 TDE 加密的数据库备份也不能在另一个没有证书的实例上进行数据库恢复，从而保证了其安全性。

SQL Server 提供了比如：DES、Triple DES、TRIPLE_DES_3KEY、RSA、RC2、RC4、DESX、AES 等加密算法，没有某种算法能适应所有要求，每种算法都有长处和短处，详细介绍请参考密码学等相关书籍。

10.1.2 数据加密和解密操作

在 SQL Server 2008 中，加密是分层级的，根层级的加密保护其子层级的加密。每一个数据库实例都拥有一个服务主密钥(Service Master Key)，这个密钥是整个实例的根密钥，在实例安装的时候自动生成，其本身由 Windows 提供的数据保护 API 进行保护，服务主密钥除了为其子节点提供加密服务之外，还用于加密一些实例级别的信息，比如实例的登录名

密码或者链接服务器的信息。

在服务主密钥之下的是数据库主密钥(Database Master Key),这个密钥由服务主密钥进行加密。这是一个数据库级别的密钥,可以用于为创建数据库级别的证书、对称或非对称密钥提供加密。每一个数据库只能有一个数据库主密钥,通过 T-SQL 语句创建,代码如下:

```
USE bankcard
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '1209QQ-1'
```

数据库主密钥由代码中所示的密码和服务主密钥共同保护。当数据库主密钥创建成功后,就可以使用这个密钥创建对称密钥、非对称密钥或证书进行数据加密了。

1. 对称加密和解密

为保护某些敏感数据,SQL Server 可以对数据库表中特定列进行加密,即使数据库被泄露,没有密钥也无法查看这些信息。

首先在 bankcard 数据库中创建一个与 account 表内容相同的 account1 表,对账号 AccNO 和密码 Password 两个列进行对称加密和解密操作,加密后会存储为二进制大对象,所以定义为 varbinary 类型。操作步骤如下:

(1) 使用数据库主密钥'1209QQ-1'加密创建对称密钥。

```
CREATE SYMMETRIC KEY PWDKEY                 --创建对称密钥 PWDKEY
WITH ALGORITHM = AES_256                    --选用的加密算法
ENCRYPTION BY PASSWORD = '1209QQ-1'         --使用数据库主密钥'加密对称密钥
```

下面就可以打开对称密钥 PWDKEY,用其加密和解密表中的数据了。

当然,为了增加安全性,也可以多加一层保护,比如用数据库主密钥加密创建的证书、对称密钥或非对称密钥再加密创建对称密钥,下面只给出一个用证书创建对称密钥的例子。

```
CREATE CERTIFICATE AcNo_Pwd                 --创建证书,证书名 AcNo_Pwd
WITH SUBJECT = 'encrypt AcNo and pwd',      --证书的主题
START_DATE = '01/01/2017',                  --证书启用日期
EXPIRY_DATE = '01/01/2019'                  --证书到期日期
GO
CREATE SYMMETRIC KEY PWDKEY
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE AcNo_Pwd          --使用证书 AcNo_Pwd 加密对称密钥
```

(2) 打开使用证书 AcNo_Pwd 加密的对称密钥。

```
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE AcNo_Pwd
```

(3) 使用对称密钥 ENCRYPTBYKEY()函数加密表中某个列的数据,语法格式为:

```
ENCRYPTBYKEY(KEY_GUID('对称密钥名'),'明文')
```

其中,KEY_GUID('对称密钥名')为加密数据所使用对称密钥的 GUID(Globally Unique Identifier)值,'明文'为要加密的数据。下面向表 account1 中插入一行数据,对账号 AccNO 和密码 Password 两个列进行对称加密。

```
INSERT INTO account1 values(ENCRYPTBYKEY(KEY_GUID('PWDKEY'),'4635121211111011112'),'133**
```

*198708150101',ENCRYPTBYKEY(KEY_GUID('PWDKEY'),'111222'),'2016-10-11','信用卡','人民币',5000,'2021-10-11')

(4) 用下面的 SELECT 语句查询 account1 表中的数据。

```
SELECT * FROM account1
```

查询结果如图 10-2 所示。

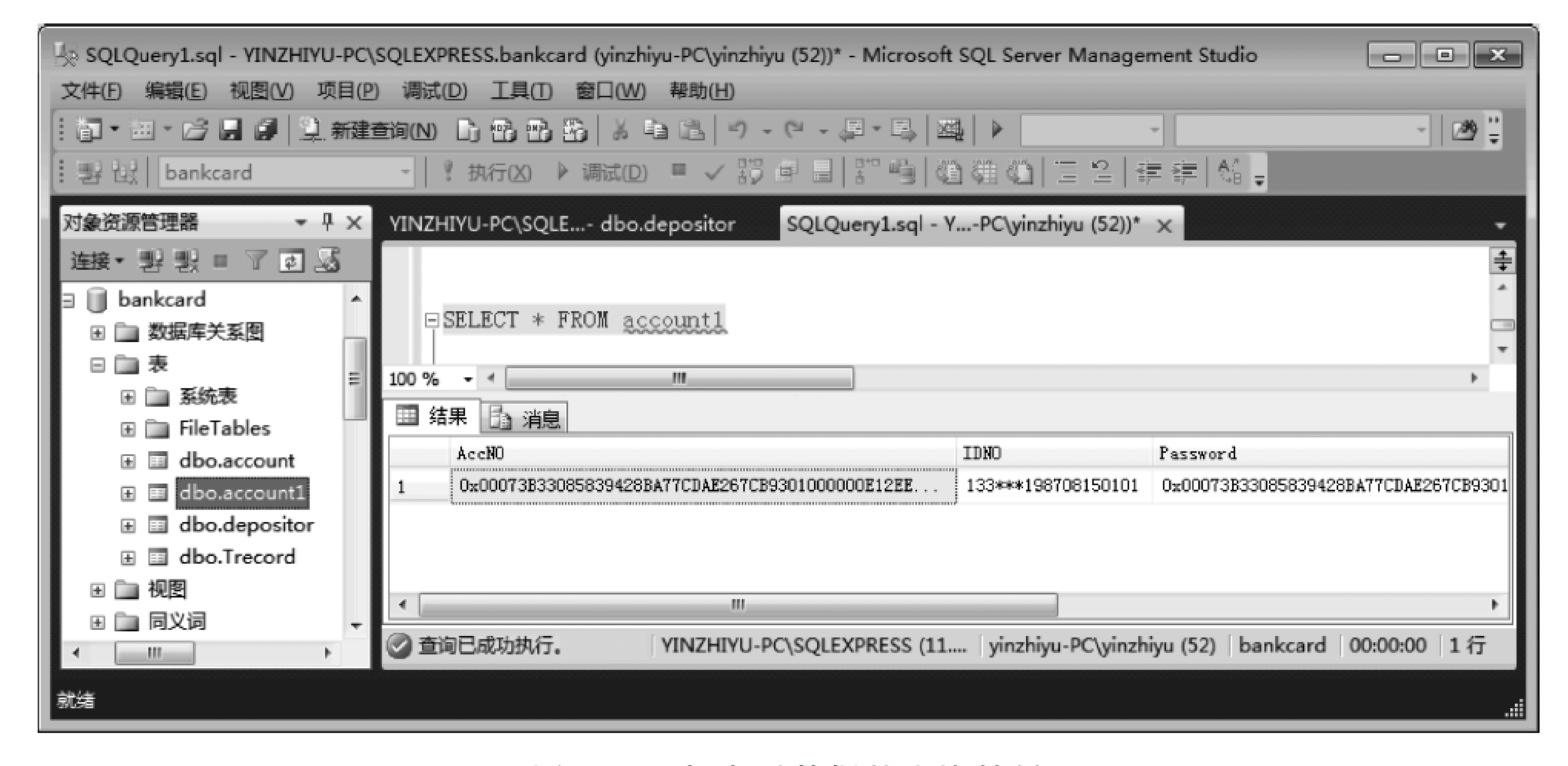


图 10-2　加密后数据的查询结果

(5) 使用对称密钥 DECRYPTBYKEY('密文')函数解密表中某个列的数据，此函数返回的是 varbinary 类型的数据，需要经过数据类型转换才能阅读。下面查询 account1 表中的数据，对账号 AccNO 和密码 Password 两个列进行对称密钥解密，并转换成字符型显示。

```
SELECT CONVERT(VARCHAR, DECRYPTBYKEY(AccNO)) AS AccNO, IDNO, CONVERT(VARCHAR, DECRYPTBYKEY
([password])) AS Password,OpenDate,CardType,MoneyType,Balance,ExpiryDate FROM account1
```

查询结果如图 10-3 所示。

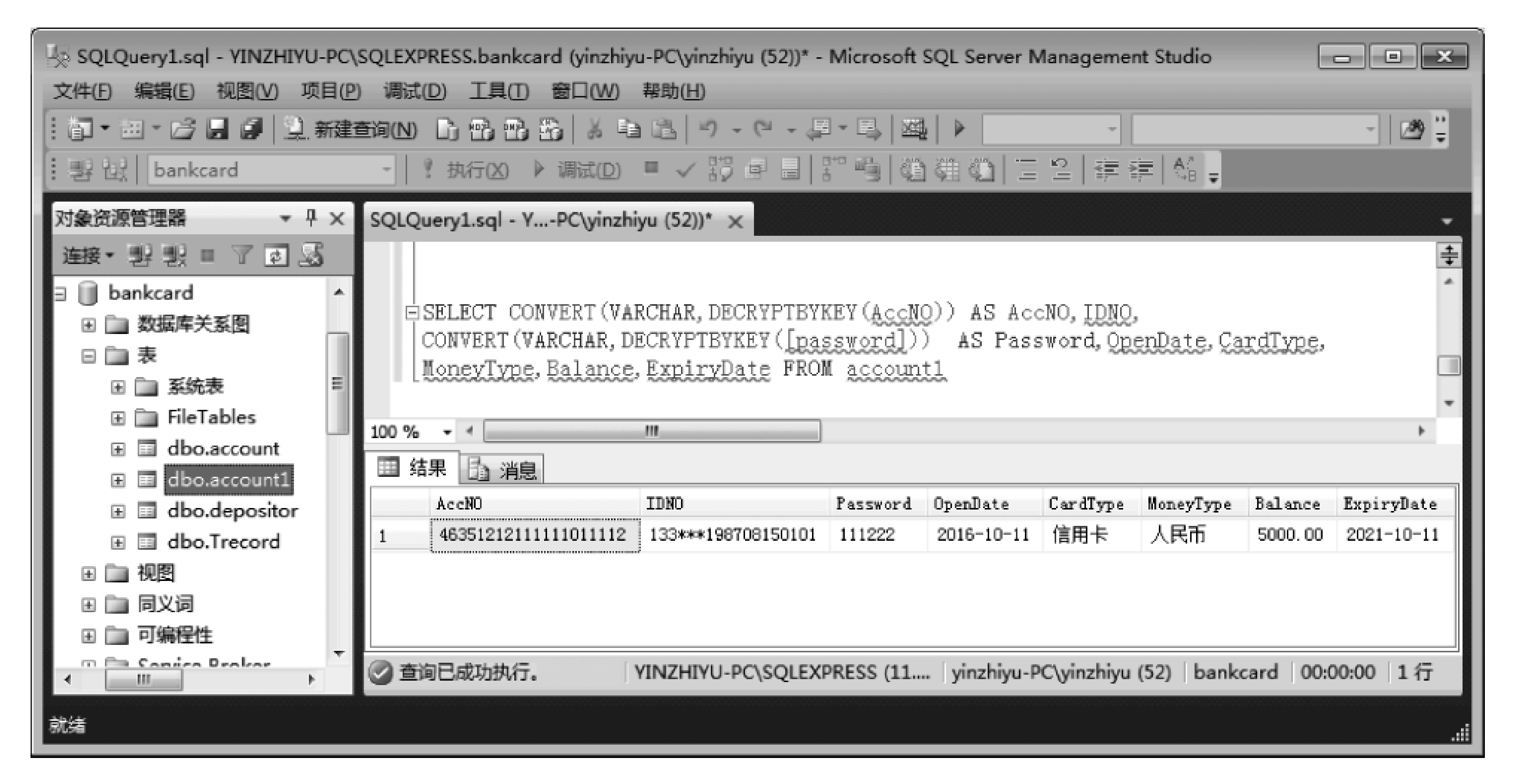


图 10-3　解密后数据的查询结果

2. 证书加密和解密

利用证书对 bankcard 数据库中 account1 表的余额 Balance 列进行加密和解密。

一个数据库只有一个主密钥，所以，无须再创建数据库主密钥，可直接使用前面创建的数据库主密钥'1209QQ-1'加密创建证书。

(1) 创建一个证书，代码如下：

```
CREATE CERTIFICATE CER_BALANCE            --创建证书,证书名 CER_BALANCE
WITH SUBJECT = 'encrypt balance',         --证书的主题
START_DATE = '01/01/2017',                --证书启用日期
EXPIRY_DATE = '01/01/2019'                --证书到期日期
GO
```

(2) 使用证书的公钥加密数据，用 EncryptByCert()函数来完成，语法格式为：

```
ENCRYPTBYCERT (CERT_ID('证书名'),'明文')
```

其中，CERT_ID('证书名')为加密数据所使用证书的 ID 值，'明文'为要加密的数据。下面向表 account1 中插入一行数据，对账号 AccNO 和密码 Password 两个列进行对称加密，对余额 Balance 列进行证书加密。

```
INSERT INTO account1 values(ENCRYPTBYKEY(KEY_GUID('PWDKEY'),'46351212111111011111'),'133***
198606150101',ENCRYPTBYKEY(KEY_GUID('PWDKEY'),'112112'),'2016-10-11','信用卡','人民币',
ENCRYPTBYCERT(CERT_ID('CER_BALANCE'),'5000'),'2021-10-11')
```

(3) 使用证书的私钥解密数据，用 DECRYPTBYCERT ('密文')函数来完成，此函数返回的是 varbinary 类型的数据，需要经过数据类型转换才能阅读。下面查询 account1 表中的数据，对账号 AccNO 和密码 Password 两个列进行对称密钥解密，对余额 Balance 列进行证书解密，并都转换成字符型显示。

```
SELECT CONVERT(VARCHAR, DECRYPTBYKEY(AccNO)) AS AccNO, IDNO, CONVERT(VARCHAR, DECRYPTBYKEY
([password])) AS Password, OpenDate, CardType, MoneyType, CONVERT(VARCHAR, DECRYPTBYCERT(CERT_
ID('CER_BALANCE'),Balance),ExpiryDate FROM account1
```

我们对 account1 表中的数据分别进行了直接查询和解密查询，查询结果如图 10-4 所示。



	AccNO	IDNO	Password	OpenDate	Car...	Mon...	Balance
1	0x0053928B698016458E8C0F026ED61...	133***198...	0x0053928B698016458E8C0F026ED61...	2016-10-11	信用卡	人民币	0x93735510181A761D4CF3186FA00990
2	0x0053928B698016458E8C0F026ED61...	133***198...	0x0053928B698016458E8C0F026ED61...	2016-10-11	信用卡	人民币	0x5F29F7CB1177459B68A66F017AB074

	AccNO	IDNO	Password	OpenDate	CardType	MoneyType	Balance	ExpiryDate
1	46351212111111011111	133***198606150101	112112	2016-10-11	信用卡	人民币	5000	2021-10-11
2	46351212111111011112	133***198705050101	111222	2016-10-11	信用卡	人民币	5000	2021-10-11

图 10-4　直接查询和解密查询结果对比

注意：使用证书加密是非对称加密操作，运行效率很低，所以最好不要在常用的数据列上使用，一般常用其加密对称密钥。

3. 使用透明数据加密和解密

在 SQL Server 2008 中引入了透明数据加密(TDE)，之所以叫透明数据加密，是因为这种加密在使用数据库的程序或用户看来，就好像没有加密一样。TDE 加密是数据库级别的，开启 TDE 的数据库的数据和日志都会被自动加密，这是由数据引擎执行的。在写入数据库时对数据进行加密，在读出数据时进行解密，客户端程序完全不用做任何操作。

TDE 使用数据加密密钥(Database Encryption Key，DEK)进行加密。DEK 存在于 Master 数据库中，由服务主密钥保护，TDE 的保护层级如图 10-5 所示。

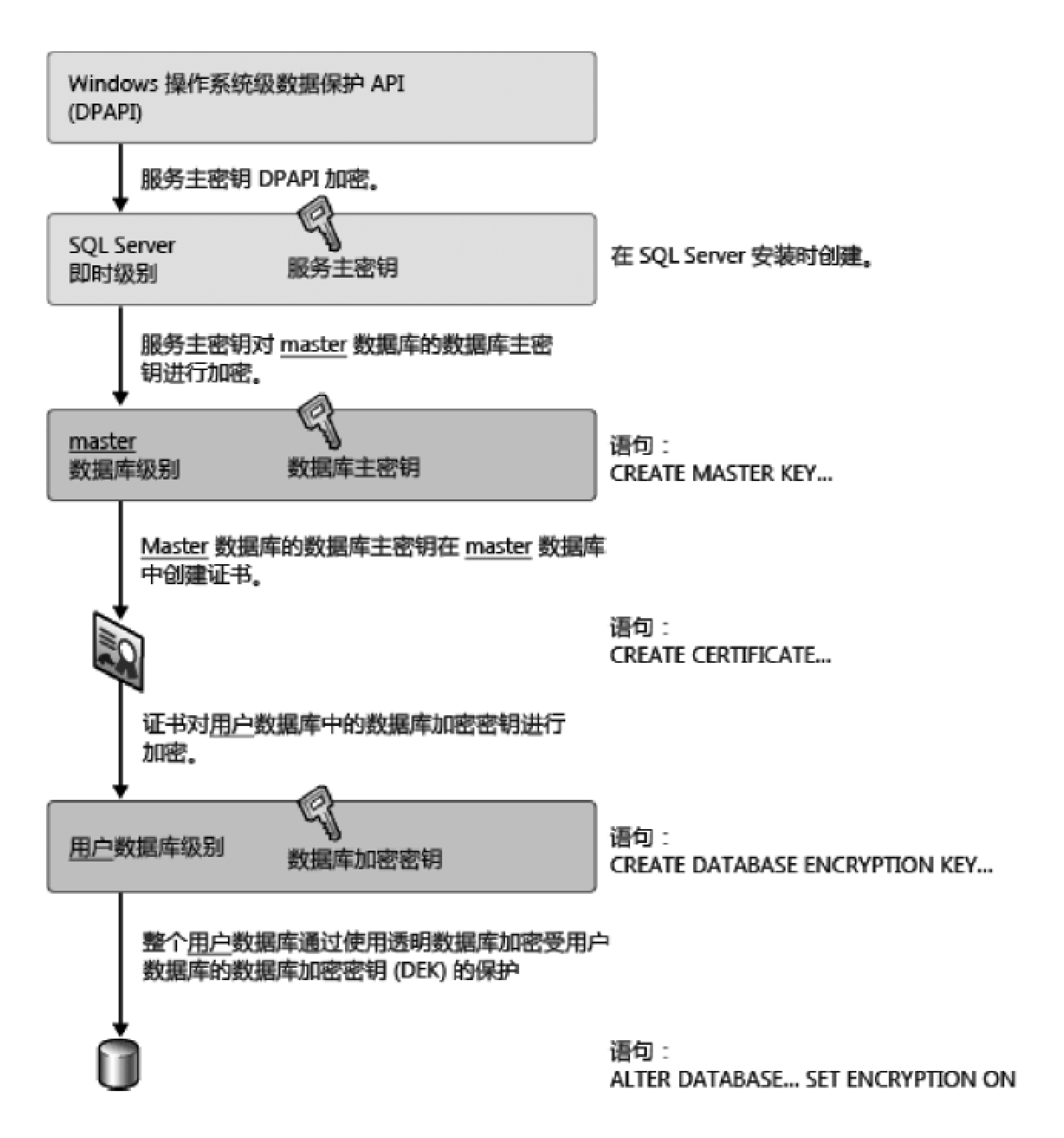


图 10-5　TDE 的加密层次

以对 teaching 数据库使用 TDE 为例，操作步骤如下：

(1) 在 master 数据库中创建数据库主密钥，代码如下：

```
USE master
CREATE MASTER KEY ENCRYPTION BY PASSWORD  = '1209QQ - 2'
```

(2) 同样在 master 数据库中创建一个证书，代码如下：

```
CREATE CERTIFICATE CER_TDE
WITH SUBJECT = 'USE TO TDE',
START_DATE = '01/01/2017',
EXPIRY_DATE = '01/01/2019'
GO
```

(3) 在 teaching 数据库中创建基于证书 CER_TDE 的数据加密密钥,代码如下:

```
USE teaching
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE CER_TDE
GO
```

(4) 修改数据库 teaching,开启 TDE,代码如下:

```
ALTER DATABASE teaching
SET ENCRYPTION ON
```

这个操作也可以通过在 SSMS 中右击需要开启 TDE 的数据库,比如 teaching,选择"任务"→"管理数据库加密"命令来进行,如图 10-6 所示。

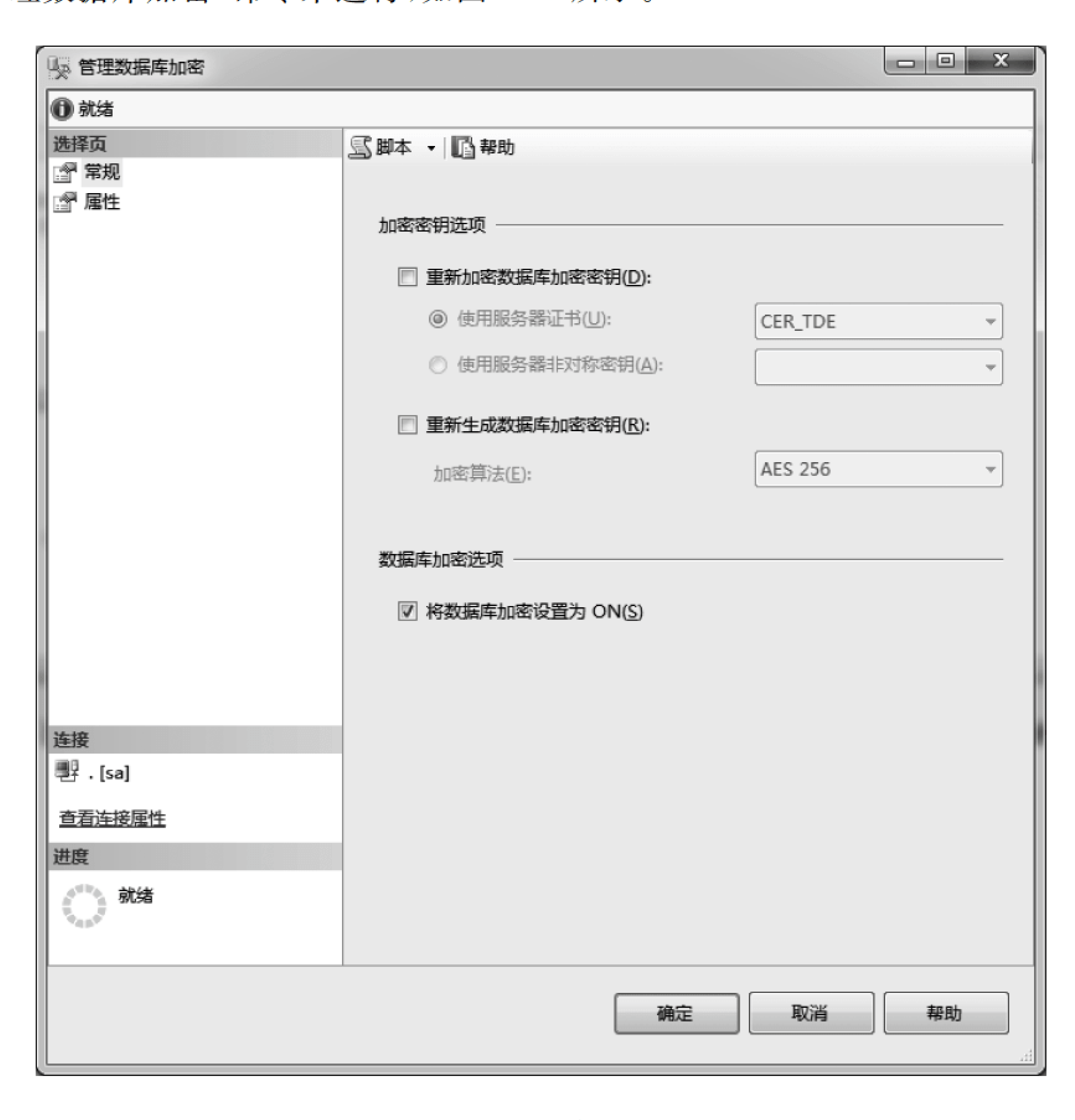


图 10-6 开启 TDE

开启 TDE 后,系统将在后台启动一个进程,进行加密扫描,直到现有数据库中所有数据加密完成。TDE 的主要作用是如果数据库备份或数据文件被非法窃取,窃取者在没有数据加密密钥的情况下是无法恢复或附加数据库的。

10.2 SQL Server 攻击与防范

黑客对 SQL Server 数据库的攻击有很多种，比如针对 SQL Server 系统本身的漏洞的攻击、针对应用程序漏洞的攻击、遍历用户密码、暴力破解超级用户 SA 的密码等等。而如何对这些攻击进行防范，比如针对 SQL Server 系统本身的漏洞的攻击就应该及时安装补丁；针对应用程序漏洞的攻击就需要管理员和程序员及时分析黑客的攻击方式，制定准确的防范策略，修改应用程序，使得其被攻击的漏洞不再出现；而数据库用户密码则需要管理员花费心思进行复杂的设置，从而尽可能地防止被黑客获取。另外，安装一个入侵检测系统或者干脆自动或手动监控日志，可以检测出大多数的黑客攻击。

以下以针对应用程序漏洞的攻击中最典型的黑客攻击手段——SQL 注入攻击为例，介绍黑客的攻击方式和防范方法。

10.2.1 SQL 注入攻击

作为一种解释型语言，SQL 在运行时由一个运行时组件解释语言代码并执行其中包含的指令。基于这种执行方式，产生了一系列叫作代码注入(Code Injection)的漏洞。

由于程序员的水平及经验参差不齐，相当大一部分程序员在编写代码的时候，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。用户可以根据程序返回的结果，获得某些他想知道的数据，这就是 SQL 注入(SQL Injection)。

SQL 注入是从正常的 WWW 端口访问，而且表面看起来跟一般的 Web 页面访问没什么区别，所以目前市面的防火墙都不会对 SQL 注入发出警报，如果管理员没有查看 IIS 日志的习惯，就可能被入侵很长时间都不会发觉。

1. SQL 注入原理

当应用程序使用输入内容来构造动态 SQL 语句以访问数据库时，就可能会发生 SQL 注入攻击。SQL 注入是通过把 SQL 命令插入到 Web 表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意 SQL 命令的目的。这可能导致攻击者使用应用程序登录到数据库中执行命令，如果应用程序使用特权过高的账户连接到数据库，这种问题会变得很严重。

许多网站程序在编写时，没有对用户输入的合法性进行判断或者程序中本身的变量处理不当，使应用程序存在安全隐患。这样，用户就可以提交一段数据库查询的代码，根据程序返回的结果，获得一些敏感的信息或者控制整个服务器，于是注入就发生了。

2. SQL 注入攻击实例

下面以某些 ASP 网站为例，进行 SQL 注入攻击的实验。

1) 攻击带有动态参数的页面

对于带有动态参数的页面，黑客都可以尝试进行 SQL 注入攻击。比如我们简单实现了这样一个页面，用于教师、学生等查询校内新闻，那么有这样一条 SQL 语句：

```
string selectStr = "select * from news where Id =" + Request.QueryString["id"];
```

正常进入此页面时，带有动态参数的网页地址 URL，比如为：

```
http://localhost:1626/news2.aspx?id = 2
```

那么后台生成的 SQL 语句为：

```
select * from news where Id = 2
```

因为这个 SQL 语句没有任何问题，所以页面会正常运行，数据会正常显示，如图 10-7 所示。

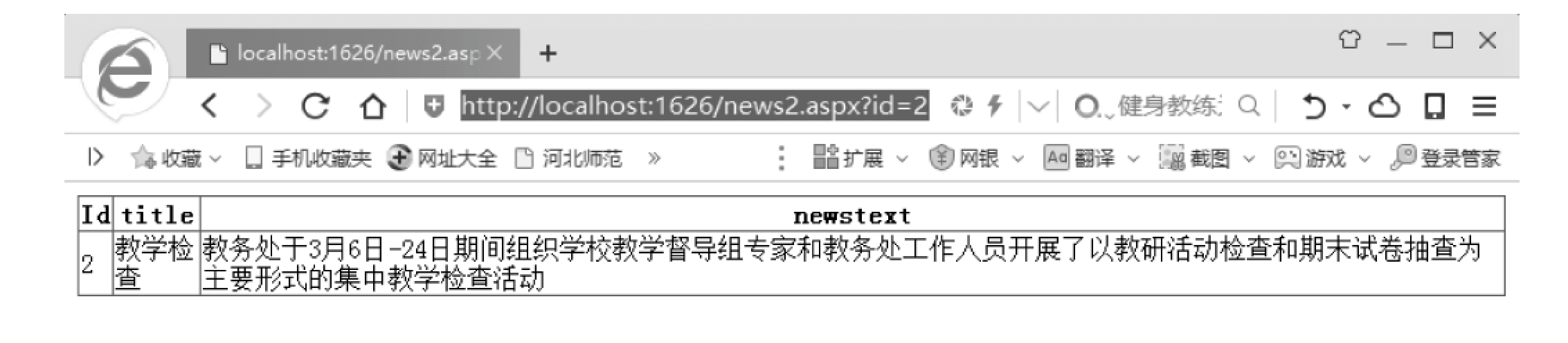




图 10-7　带有动态参数的页面

下面黑客可以开始测试这里是否为 SQL 注入点。即在 URL 后加入其他信息，通过网页的运行结果，判断其是否为注入点。

首先，可以在网页地址栏输入：

```
http://localhost:1626/news2.aspx?id = 2 and 1 = 1
```

那么后台生成的 SQL 语句为：

```
select * from news where Id = 2 and 1 = 1
```

很显然，这个 SQL 语句也没有任何问题，所以页面也会正常运行，数据正常显示，如图 10-8 所示。



图 10-8　URL 后加入 and 1=1 的情况

然后，用户可以在网页地址栏输入：

```
http://localhost:1626/news2.aspx?id = 2 and 1 = 2
```

那么后台生成的 SQL 语句为：

```
select * from news where Id = 2 and 1 = 2
```

这个 SQL 语句在语法上是没有任何问题的，但查询条件不成立，所以页面上会显示数据不存在或抛出异常，我们设计的页面运行结果为显示空白，如图 10-9 所示。

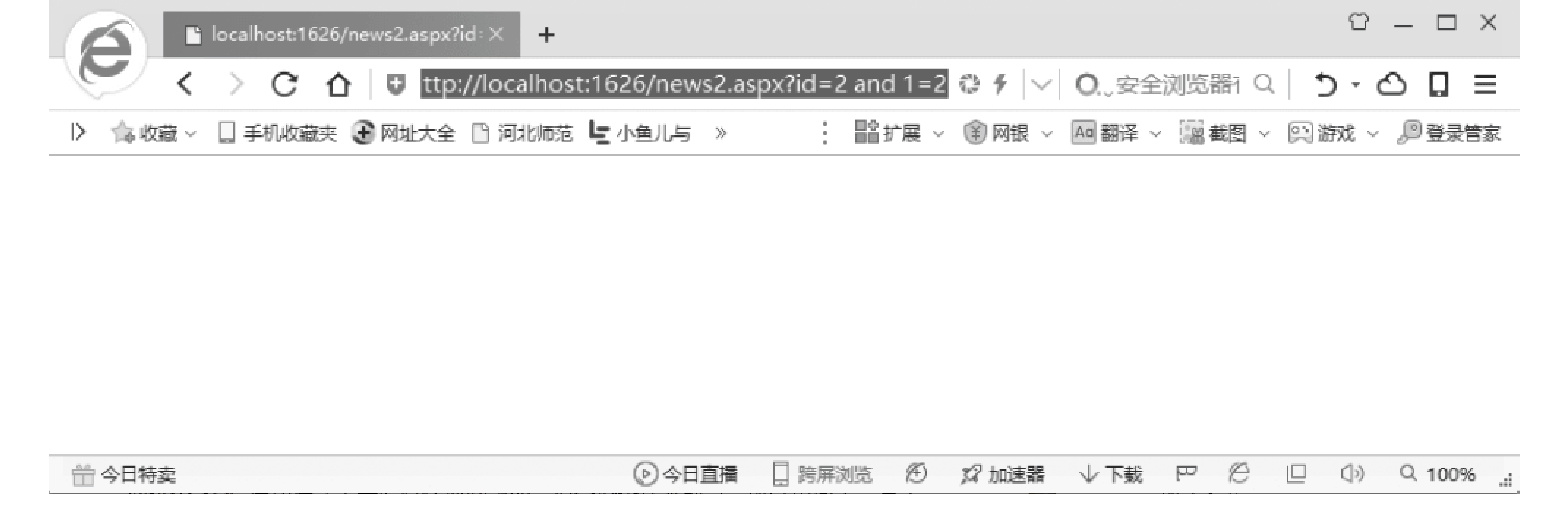


图 10-9　URL 后加入 and 1=2 的情况

根据两次测试的结果，可以断定在 URL 后是可以加入其他信息的，所以这里就是 SQL 注入点。而如果两次测试的运行结果相同，都出现程序定义的错误提示或提示类型转换时出错等非正常运行结果，那么这里就不是 SQL 注入点。

发现注入点后，黑客可以通过某个 SQL 注入工具来获取数据库中的数据、上传木马等等。

如果应用程序具有数据库管理员的权限，那么黑客也可以直接在 URL 后加入对数据库表中数据具有破坏性的语句。比如，用户如果在网页地址栏输入：

```
http://localhost:1626/news2.aspx?id = 2 ;drop table news
```

就会将新闻表 news 从数据库中删除。

2）登录时绕过用户认证

比如“教学管理系统”的登录页面后台有这样一段代码：

```
string userName = TextBox1.Text.Trim();              /* 用户名 */
string userPwd = TextBox2.Text.Trim();               /* 密码 */
string userRole = RadioButtonList1.SelectedValue;    /* 用户角色，教师、学生或管理员 */
string selectStr = "";
switch(userRole)
{  case "0" :/* 用户角色为教师 */
   selectStr = "select * from Teacher where 教师号 = '" + userName + "' and 密码 = '" + userPwd + "'";
   break;
   case "1":/* 用户角色为学生 */
   selectStr = "select * from Student where 学号 = '" + userName + "' and 密码 = '" +
userPwd + "'";
   break;
   case "2":/* 用户角色为管理员 */
   selectStr = "select * from Manager where 管理员账号 = '" + userName + "' and 密码 = '" +
userPwd + "'";
   break;
}
```

如果用户正常登录，为了实验更清楚，我们提前把密码框设置为可见，界面如图 10-10 所示。

图 10-10 用户正常登录

后台生成的 SQL 语句为：

```
select * from Manager where 管理员账号 = '20001' and 密码 = '111111'
```

则用户验证通过，登录成功。

如果非法用户进入登录界面，知道正确的用户名但不知道密码，所以可以如图 10-11 所示进行输入。

图 10-11 非法的用户登录

那么后台生成的 SQL 语句为：

```
select * from Manager where 管理员账号 = '20001' -- 'and 密码 = '11'
```

语句后半部分的查询条件被注释掉了，所以只需通过用户名就可以成功登录。

10.2.2 SQL 注入攻击的防范

对于 SQL 注入攻击，可以使用相应的方法进行防范，一般可以采用在数据库服务器端编写存储过程和在应用程序开发过程中过滤敏感字符等。

1. 使用存储过程

对于 10.2.1 节介绍的对带有动态参数的页面的攻击，可以利用存储过程进行防范。比如查询校内新闻的页面，可以将其中的查询语句编写成一个存储过程，而在应用程序中通过参数传递来执行存储过程。

创建存储过程的代码如下：

```
CREATE PROCEDURE [dbo].querynews
    @id int
AS
    SELECT * FROM news WHERE Id = @id
```

应用程序中的 SQL 语句如下：

```
string selectStr = "exec querynews" + Request.QueryString["id"];
```

正常进入此页面时，带有动态参数的网页地址 URL，比如为：

```
http://localhost:1626/news2.aspx?id = 2
```

那么后台生成的 SQL 语句为：

```
exec querynews 2
```

那么此页面会正常运行，运行结果如图 10-7 所示。

黑客判断其是否为注入点时，同样可以在网页地址栏输入：

```
http://localhost:1626/news2.aspx?id = 2 and 1 = 1
```

那么后台生成的 SQL 语句为：

```
exec querynews 2 and 1 = 1
```

然后在网页地址栏输入：

```
http://localhost:1626/news2.aspx?id = 2 and 1 = 1
```

那么后台生成的 SQL 语句为：

```
exec querynews 2 and 1 = 2
```

而两次测试的运行结果相同，都出现如图 10-12 所示的错误提示，所以无法进行 SQL 注入攻击。

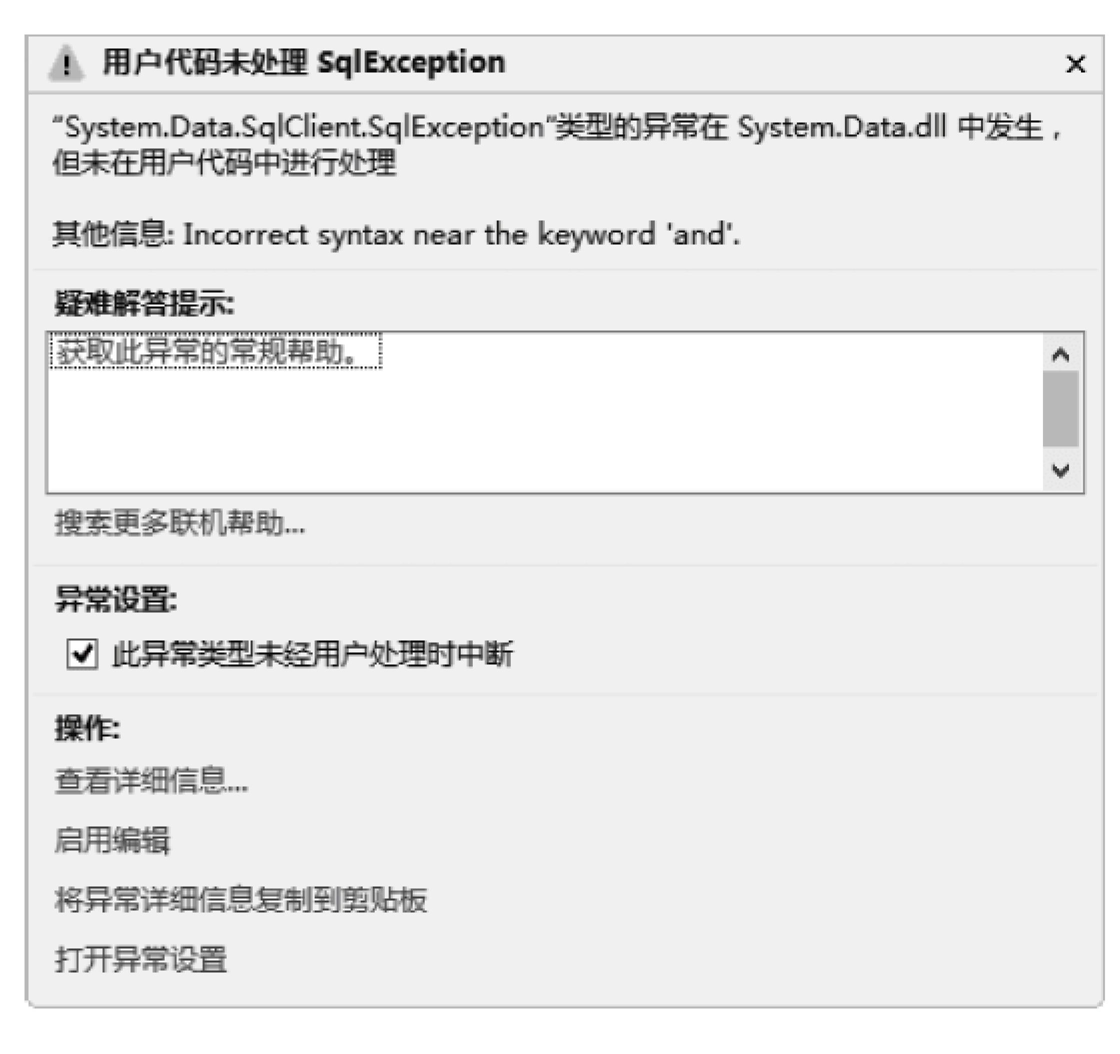


图 10-12　SQL 注入点测试失败

2. 使用单条件查询语句

对于10.2.1节中登录时绕过用户认证的SQL注入攻击，利用存储过程无法进行防范，因为登录时的查询语句中包含多个查询条件，所以就要向存储过程传递多个参数，涉及动态拼接的SQL语句，仍然可以进行SQL注入攻击。

我们可以将登录时的查询语句改成单条件查询。首先用用户的登录名作为查询条件，查询此用户是否存在，如果不存在，则直接提示“用户名输入错误”；如果存在，则判断其密码是否正确，这样的查询就使得黑客无法进行SQL注入攻击。

这里省略其他角色，只给出学生登录时的程序代码：

```
SqlConnection conn = new SqlConnection(s);            /*建立数据库连接*/
string userName = TextBox1.Text.Trim();               /*用户的登录名*/
string userPwd = TextBox2.Text.Trim();                /*用户的密码*/
string selectStr = "";
selectStr = "select * from student where 学号 = '" + userName + "'";
SqlCommand cmd = new SqlCommand(selectStr,conn);      /*建立命令对象*/
 conn.Open();
 SqlDataReader dr = cmd.ExecuteReader();              /*执行命令，读取数据*/
    if (dr.Read())
     {
       string pwd = (string)dr[1];
       if(pwd == userPwd)                             /*判断密码是否正确*/
         Response.Redirect("student.aspx");
       else
         Label3.Text = "密码错误!";
     }
   else
   Label3.Text = "用户名错误!";
```

3. 过滤敏感字符

对于带有动态参数的页面的SQL注入攻击和登录时绕过用户认证，都可以通过使用过滤注释符号“--”、单引号“'”、and、or等敏感字符的方法进行防范。

敏感字符的过滤方式完全在应用程序中实现，可以在发现敏感字符时在页面显示如“您提交的内容中含有非法字符”这样的提示，而不显示网页实际内容；或者将敏感字符用其他字符代替。

过滤敏感字符的编程实现起来很简单，这里不进行详细介绍，读者可以在应用程序开发的相关课程中进行学习。

另外，可以在应用程序中进行输入合法性检查、限制输入数据的长度和范围等，来提高系统的安全性，也方便用户操作。还有最重要的一点：一定要对应用程序使用的账号做严格的权限管理，并且尽量不要将管理员账号给应用程序使用。

习 题 10

1. 简述4对加密和解密的内置函数。
2. 简述透明数据加密的作用。

3. 在 teaching 数据库的学生表中添加一个“密码”列，varbinary(500)类型，采用对称密钥加密存储数据，并用对称密钥解密显示数据。

4. 什么是 SQL 注入攻击？

5. 编写一个存储过程，用于实现学生通过密码登录教务系统，其中密码列为 3 题中对称密钥加密存储。要求通过学号和密码验证用户身份，显示学号和密码正确的学生姓名，此过程要能够防范 SQL 注入攻击。

第11章　数据库文件的安全备份与恢复

避免数据丢失是数据库管理员需要面对的最关键的问题之一。尽管在 SQL Server 2008 中采取了许多措施来保证数据库的安全性和完整性，但故障仍不可避免，仍会影响甚至破坏数据库，造成数据丢失。同时还存在其他一些可能造成数据丢失的因素，例如用户的操作失误、蓄意破坏、病毒攻击和自然界的不可抗力等。因此，对数据库文件的安全备份和恢复是非常重要的数据库安全保护措施。

本章主要介绍数据库备份和还原的概念及其重要性，SQL Server 2008 对数据库进行备份和还原操作的方法，数据库的分离和附加方法，数据库快照、数据库镜像的概念和操作。

11.1　数据库的备份与还原

数据库的备份与还原是数据库文件的安全备份和恢复中最简单也是最常用的操作。SQL Server 2008 制定了一个良好的备份还原策略，定期将数据库进行备份以保护数据库，以便在事故发生后还原数据库。

11.1.1　数据库备份概述

对于计算机用户来说，对一些重要文件、资料定期进行备份是一种良好的习惯。如果出现突发情况，比如系统崩溃、系统遭受病毒攻击等，使得原先的文件遭到破坏以至于全部丢失，启用文件备份，就可以节省大量的时间和精力。

数据库备份就是在某种介质上(磁盘、磁带等)创建完整数据库(或者其中一部分)的副本，并将所有的数据项都复制到备份集，以便在数据库遭到破坏时能够恢复数据库。

对 SQL Server 2008 数据库或事务日志进行备份，就是记录在进行备份这一操作时数据库中所有数据的状态，以便在数据库遭到破坏时能够及时地将其还原。执行备份操作必须拥有对数据库备份的权限许可，SQL Server 2008 只允许系统管理员、数据库所有者和数据库备份执行者备份数据库。

SQL Server 2008 提供了高性能的备份和还原功能以及保护手段，以保护存储在 SQL Server 2008 数据库中的关键数据。通过适当的备份，可以使用户能够在发生多种可能的故障后恢复数据，这些故障主要包括：系统故障；用户错误，例如，误删除了某个表或某些数据；硬件故障，例如，磁盘驱动器损坏；自然灾害。

1. 数据库备份计划

创建备份的目的是为了恢复已损坏的数据库。但是，备份和还原数据需要在特定的环境中进行，并且必须使用一定的资源。因此，在备份数据库之前，需要对备份内容、备份频率

以及数据备份存储介质等进行计划。

1）备份内容

备份内容主要包括系统数据库、用户数据库和事务日志。

（1）系统数据库记录了 SQL Server 系统配置参数、用户资料以及所有用户数据库等重要信息，主要包括 master、msdb 和 model 数据库。

（2）用户数据库中存储了用户的数据。由于用户数据库具有很强的区别性，即每个用户数据库之间的数据一般都有很大差异，所以对用户数据库的备份更为重要。

（3）事务日志记录了用户对数据库中数据的各种操作，平时系统会自动管理和维护所有的数据库事务日志。相比数据库备份，事务日志备份所需要的时间较少，但是还原需要的时间较多。

2）备份频率

数据库备份频率一般取决于修改数据库的频繁程度，以及一旦出现意外丢失的工作量的大小，还有发生意外丢失数据的可能性大小。

一般来说，在正常使用阶段，对系统数据库的修改不会十分频繁，所以对系统数据库的备份也不需要十分频繁，只需要在执行某些语句或存储过程导致 SQL Server 2008 对系统数据库进行了修改的时候备份。

当在用户数据库中执行了加入数据、创建索引等操作时，应该对用户数据库进行备份，此外，如果清除了事务日志，也应该备份数据库。

3）备份存储介质

常用的备份存储介质包括硬盘、磁带和命令管道等。具体使用哪一种介质，要考虑用户的成本承受能力、数据的重要程度、用户的现有资源等因素。在备份中使用的介质确定以后，一定要保持介质的持续性，一般不要轻易地改变。

4）其他计划

（1）确定备份工作的负责人。

备份负责人负责备份的日常执行工作，并且要经常进行检查和督促。这样，可以明确责任，确保备份工作得到人力保障。

（2）确定使用在线备份还是脱机备份。

在线备份就是动态备份，允许用户继续使用数据库。脱机备份就是在备份时，不允许用户使用数据库。虽然备份是动态的，但是用户的操作会影响数据库备份的速度。

（3）确定是否使用备份服务器。

在备份时，如果有条件最好使用备份服务器，这样可以在系统出现故障时，迅速还原系统的正常工作。当然，使用备份服务器会增大备份的成本。

（4）确定备份存储的地方。

备份是非常重要的内容，一定要保存在安全的地方。在保存备份时应该实行异地存放，并且每套备份的内容应该有两份以上。

（5）确定备份存储的期限。

对于一般性的业务数据可以确定一个比较短的期限，但是对于重要的业务数据，需要确定一个比较长的期限。期限愈长，需要的备份介质就愈多，备份成本也随之增大。

总之，备份应该按照需要经常进行，并进行有效的数据管理。SQL Server 2008 备份可

以在数据库使用时进行，但是一般在非高峰活动时备份效率更高。另外，备份是一种十分耗费时间和资源的操作，不能频繁操作。应该根据数据库的使用情况确定一个适当的备份周期。

2. 数据库备份的类型

在 SQL Server 系统中，有 4 种备份类型，分别如下所示：

（1）完整数据库备份。

（2）差异数据库备份。

（3）事务日志备份。

（4）数据库文件或文件组备份。

下面分别介绍这 4 种类型的备份。

1）完整数据库备份

完整数据库备份将备份整个数据库，包括事务日志部分（以便可以恢复整个备份）。完整数据库备份代表备份完成时的数据库，通过包括在备份中的事务日志，可以使用备份恢复到备份完成时的数据库。

创建完整数据库备份是单一操作，通常会安排该操作定期发生。如果数据库主要是只进行读操作，那么完整数据库备份能有效地防止数据丢失。

完整数据库备份易于使用。因为完整数据库备份包含数据库中的所有数据，所以对于可以快速备份的小数据库而言，最佳方法就是使用完整数据库备份。但是，随着数据库的不断增大，完整备份需花费更多时间才能完成，并且需要更多的存储空间。因此，对于大型数据库而言，可以用差异数据库备份来补充完整数据库备份。

2）差异数据库备份

差异数据库备份只备份自上一次完整数据库备份发生改变的内容和在差异数据库备份过程中所发生的所有活动。差异数据库备份基于以前的完整数据库备份，因此，这样的完整数据库备份称为“基准备份”。差异备份比完整备份更小、更快，可以简化频繁的备份操作，减少数据丢失的风险。为了减少还原频繁修改数据库的备份时间，可以执行差异备份。

如果数据库的某个子集比该数据库的其余部分修改得更为频繁，则差异数据库备份特别有用。在这些情况下，使用差异数据库备份，可以频繁执行备份，并且不会产生完整数据库备份的开销。

对于大型数据库，完整数据库备份需要大量磁盘空间。为了节省时间和磁盘空间，可以在一次完整数据库备份后安排多次差异备份。每次连续的差异数据库备份都大于前一次备份，这就需要更长的备份时间、还原时间和更大的空间。因此，可以定期执行新的完整备份以提供新的差异基准。

当使用差异数据库备份时，最好遵循以下原则：

在每次完整数据库备份后，定期安排差异数据库备份。例如，可以每 4 小时执行一次差异数据库备份，对于活动性较高的系统，此频率也可以更高。

在确保差异备份不会太大的情况下，定期安排新的完整数据库备份。例如，可以每周备份一次完整数据库。

3）事务日志备份

备份事务日志可以记录数据库的更改，但前提是在执行了完整数据库备份之后。可以

使用事务日志备份将数据库恢复到特定的即时点(如输入多余数据前的那一点)或恢复到故障点。

恢复事务日志备份时,SQL Server 2008 重做事务日志中记录的所有更改。当 SQL Server 2008 到达事务日志的最后时,已重新创建了与开始执行备份操作的那一刻完全相同的数据库状态。如果数据库已经恢复,则 SQL Server 2008 将回滚备份操作开始时尚未完成的所有事务。

一般情况下,事务日志备份比数据库备份使用的资源少,因此可以比数据库备份更经常地创建事务日志备份,经常备份将减少丢失数据的危险。

如图 11-1 所示为基于完整恢复模式(详见 11.1.2 节)下的 1 个完整备份+N 个连续的事务日志备份的策略。如果中间的日志备份 02 删除或者损坏,则数据库只能恢复到日志备份 01 的即时点。

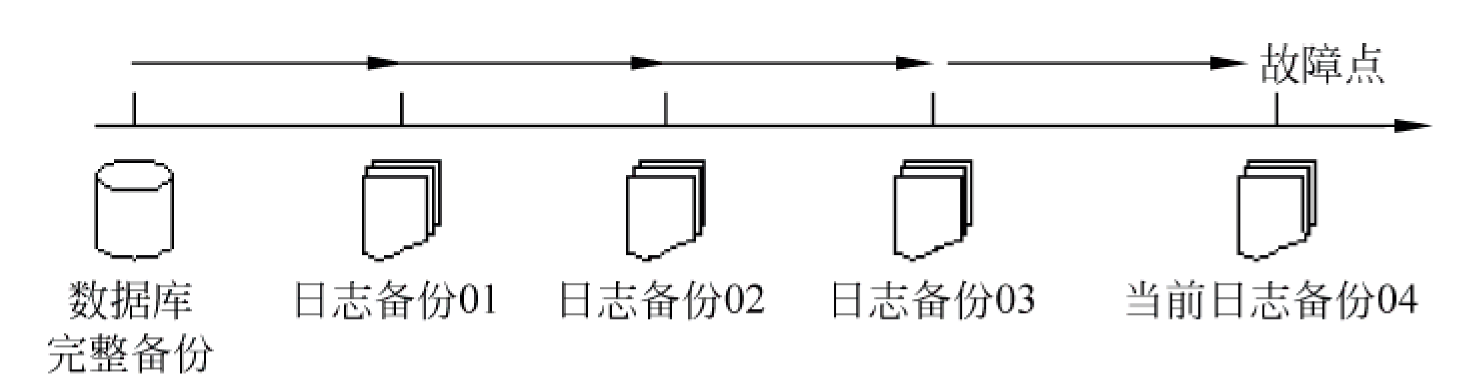


图 11-1 事务日志备份与恢复原理

假如日志备份 01、02 和 03 都是完整的,那么在恢复时,先恢复数据库完整备份,然后依次恢复日志备份 01、02 和 03。如果要恢复到故障点,就需要看数据库的当前日志是否完整,如果是完整的,可以做一个当前日志的备份,然后依次恢复到日志备份 04 就可以了。

4) 数据库文件或文件组备份

对超大型数据库执行完全数据库备份是不可行的,可以执行数据库文件或文件组备份。备份文件或文件组时,可以只备份 FILE 或 FILEGROUP 选项中指定的数据库文件。在备份数据库文件或文件组时应考虑以下几点。

(1) 必须指定逻辑文件或文件组。

(2) 必须执行事务日志备份,使还原的文件与数据库的其他部分相一致。

(3) 最多可以指定 16 个文件或文件组。

(4) 应制定轮流备份每个文件的计划。

11.1.2 数据库还原概述

备份是还原数据库最容易和最能防止意外的有效方法。没有备份,所有的数据都可能会丢失,而且将造成不可挽回的损失,这时就不得不从源头重建数据;有了备份,万一数据库被损坏,就可以使用备份来还原数据库。

1. 数据库还原策略

还原数据库是一个装载数据库的备份,然后应用事务日志重建的过程,这是数据库管理员另一项非常重要的工作。应用事务日志之后,数据库就会回到最后一次事务日志备份之前的状况。在数据库备份之前,应该检查数据库中数据的一致性,这样才能保证顺利地还原数据库备份。在数据库的还原过程中,用户不能进入数据库,当数据库被还原后,数据库中的所有数据都被替换掉。数据库备份是在正常情况下进行的,而数据库还原是在诸如硬件

故障、软件故障或误操作等非正常的状态下进行的，因而其工作更加重要和复杂。

数据还原策略认为所有的数据库一定会在它们的生命周期的某一时刻需要还原。数据库管理员职责中很重要的部分就是将数据还原的频率降到最低，并在数据库遭到破坏之前进行监视，预计各种形式的潜在风险所能造成的破坏，并针对具体情况制定恢复计划，在破坏发生时及时地恢复数据库。

还原方案从一个或多个备份中还原数据，并在还原最后一个备份后恢复数据库。如果数据库做过完全备份和事务日志备份，那么还原它是很容易的，倘若保持着连续的事务日志，就能快速地重新构造和建立数据库。还原数据库是一个装载最近备份的数据库和应用事务日志来重建数据库到失效点的过程。定点还原可以把数据库还原到一个固定的时间点，这种选项仅适用于事务日志备份。当还原事务日志备份时，必须按照它们建造的顺序还原。

在还原一个失效的数据库之前，调查失效背后的原因是很重要的。如果数据库的损坏是由介质错误引起的，那么就需要替换失败的介质。倘若是由于用户的问题而引起的，那么就需要针对发生的问题和今后如何避免采取相应的对策。如果是由系统故障或自然灾害引起的，那么就只能具体问题具体分析，根据损害的程度采取相应的对策。比如死机，只需重新启动操作系统和 SQL Server 服务器，重做没有提交的事务；如果数据库损坏，可以通过备份还原；而如果介质损坏，只能替换；等等。

2. 数据库恢复模式

数据库的恢复模式是数据库遭到破坏时还原数据库中数据的数据存储方式，它与可用性、性能、磁盘空间等因素相关。备份和还原操作是在“恢复模式”下进行的，恢复模式是一个数据库属性，它用于控制数据库备份和还原操作基本行为。

每一种恢复模式都按照不同的方式维护数据库中的数据和日志。Microsoft SQL Server 2008 系统提供了 3 种数据库的恢复模式：

(1) 完整恢复模式。

(2) 简单恢复模式。

(3) 大容量日志恢复模式。

1) 完整恢复模式

完整恢复模式是等级最高的数据库恢复模式。在完整恢复模式中，对数据库的所有的操作都记录在数据库的事务日志中。即使那些大容量数据操作和创建索引的操作，也都被记录在了数据库的事务日志中。当数据库遭到破坏之后，可以使用该数据库的事务日志迅速还原数据库。

在完整恢复模式中，由于事务日志记录了数据库的所有变化，所以可以使用事务日志将数据库还原到任意的时刻点。但是，这种恢复模式耗费大量的磁盘空间。除非是那种事务日志非常重要的数据库备份策略，一般不使用这种恢复模式。

这种恢复模式的特点是：

(1) 允许将数据库还原到故障点状态。

(2) 数据库可以进行 4 种备份方式中的任何一种。

(3) 可以还原到即时点。

这种模式的优点是数据丢失或损坏不导致工作损失，可还原到即时点。但所有修改都

记录在日志中，发生某些大容量操作时日志文件增长太快。如果系统符合下列任何要求，则使用完整恢复模式：

(1) 用户必须能够恢复所有数据。

(2) 数据库包含多个文件组，并且希望逐段还原读写辅助文件组(以及只读文件组)。

(3) 必须能够恢复到故障点。

2) 简单恢复模式

简单恢复模式简略地记录大多数事务，所记录的信息只是为了确保在系统崩溃或还原数据备份之后数据库的一致性。

对于那些规模比较小的数据库或数据不经常改变的数据库来说，可以使用简单恢复模式。当使用简单恢复模式时，可以通过执行完全数据库备份和差异数据库备份来还原数据库，数据库只能还原到执行备份操作的时刻点。执行备份操作之后的所有数据修改都会丢失并且需要重建。

这种恢复模式的特点是：

(1) 允许将数据库还原到最新的备份。

(2) 数据库只能进行完全数据库备份和差异备份，不能进行事务日志备份以及文件和文件组备份。

(3) 不能还原到某个即时点。

这种模式的优点是所有操作使用最少的日志空间记录，节省空间，恢复模式最简单。如果系统符合下列所有要求，则使用简单恢复模式：

(1) 丢失日志中的一些数据无关紧要。

(2) 无论何时还原主文件组，用户都希望始终还原读写辅助文件组(如果有)。

(3) 是否备份事务日志无所谓，只需要完整差异备份。

(4) 不在乎无法恢复到故障点以及丢失从上次备份到发生故障时之间的任何更新。

3) 大容量日志恢复模式

就像完整恢复模式一样，大容量日志恢复模式也使用数据库备份和日志备份来还原数据库。但是，在使用了大容量日志恢复模式的数据库中，其事务日志耗费的磁盘空间远远小于使用完整恢复模式的数据库的事务日志。

此模式简略地记录大多数大容量操作(例如，索引创建和大容量加载)，完整地记录其他事务。大容量日志恢复提高了大容量操作的性能，常用作完整恢复模式的补充。

这种恢复模式的特点是：

(1) 还原允许大容量日志记录的操作。

(2) 数据库可以进行 4 种备份方式中的任何一种。

(3) 不能还原到某个即时点。

这种模式的优点是对大容量操作使用最少的日志记录，节省日志空间；缺点是丧失了恢复到即时点的功能，如非特别需要，不建议使用此模式。

在 Microsoft SQL Server 2008 系统中有两种设置数据库恢复模式的方式，即 SSMS 和 ALTER DATABASE 语句。

这里主要介绍前一种方法：在 SSMS 环境下，选中将要设置恢复模式的数据库，右击数据库，从弹出的快捷菜单中选择“属性”命令，将出现如图 11-2 所示的“数据库属性”对话框。

在该对话框的“选项”选择页中，从“恢复模式”下拉列表中选择恢复模式。



图 11-2　设置数据库恢复模式

简单恢复模式同时支持数据库备份和文件备份，但不支持事务日志备份。备份非常易于管理，因为始终不会备份事务日志。但是，如果没有日志备份，数据库只能还原到最近数据备份的末尾。如果操作失败，则在最近数据备份之后所做的更新便会全部丢失。

在完整恢复模式和大容量日志恢复模式下，差异数据库备份将最大限度地减少在还原数据库时回滚事务日志备份所需的时间。

事务日志备份只能与完整恢复模式和大容量日志记录恢复模式一起使用。在简单模型下，事务日志有可能被破坏，所以事务日志备份可能不连续，不连续的事务日志备份没有意义，因为基于日志的恢复要求日志是连续的。

11.1.3　数据库备份操作

在 Microsoft SQL Server 2008 中，数据库备份操作有两种方式：在 SSMS 中使用界面备份数据库和使用 T-SQL 语句备份数据库。

1. 在 SSMS 中使用界面备份数据库

【例 11-1】 在 SSMS 的“对象资源管理器”中创建 teaching 的完整数据库备份，操作步骤如下：

(1) 在“对象资源管理器”中展开 teaching 数据库。

(2) 右击 teaching，在弹出的快捷菜单中选择“任务”→“备份”命令，弹出“备份数据库-

teaching”窗口，如图 11-3 所示。

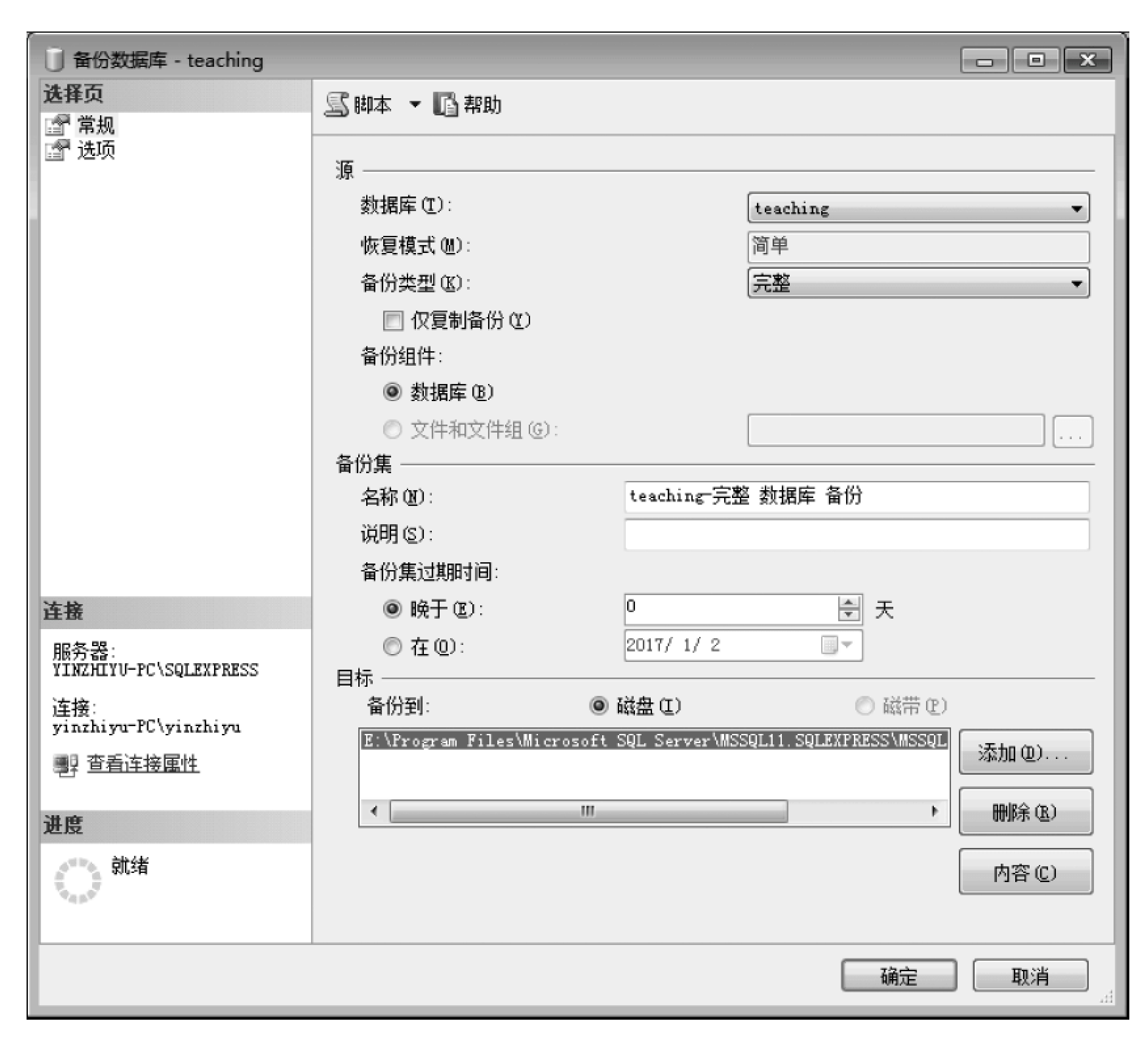


图 11-3　备份数据库-teaching

(3) 在“数据库”下拉列表框中选择 teaching 作为准备备份的数据库。在“备份类型”下拉列表框中选择需要的类型，这是第一次备份，选择“完整”选项，在“名称”文本框中输入要备份的名称。

(4) 由于没有磁带设备，所以只能备份到“磁盘”。单击“添加”按钮，重新选择路径并给出文件名，最后单击“确定”按钮，如图 11-4 所示。

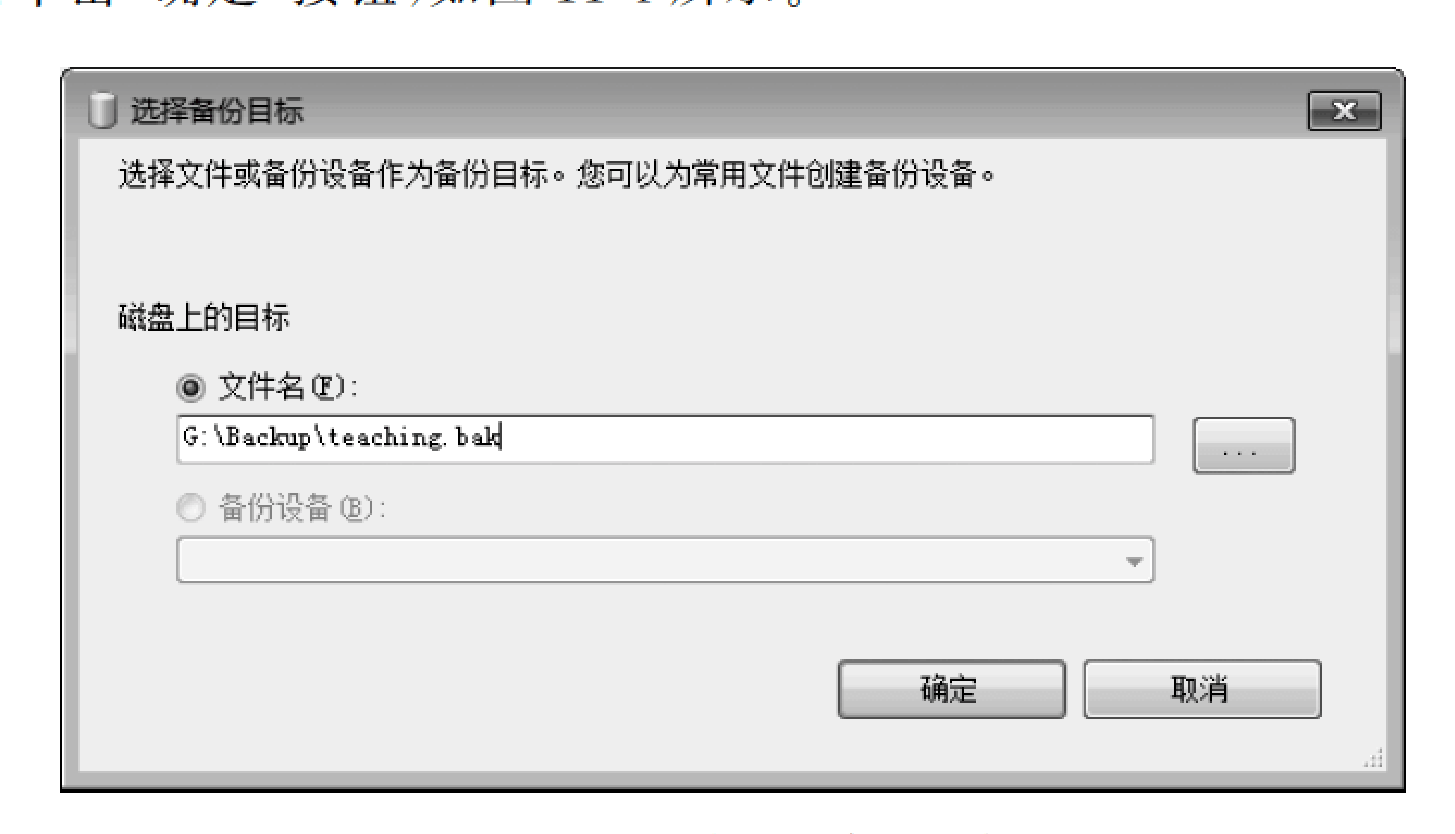


图 11-4　选择路径和命名文件名

(5) 单击“备份数据库-teaching”窗口左上角的“选项”选项，对“备份到现有介质集”选项进行设置，此选项的含义是备份介质的现有内容被新备份重写。在“备份到现有介质集”

选项组中含有两个选项："追加到现有备份集"和"覆盖所有现有备份集"。其中"追加到现有备份集"是介质上以前的内容保持不变，新的备份在介质上次备份的结尾处写入。"覆盖所有现有备份集"是重写备份设备中任何现有的备份。此处选中"追加到现有备份集"单选按钮，单击"确定"按钮，数据备份完成，如图 11-5 所示。



图 11-5　数据库 teaching 备份完成

2. 使用 T-SQL 语句备份数据库

使用 T-SQL 语句备份数据库的基本语法格式如下：

```
BACKUP DATABASE { database_name | @database_name_var }
TO < backup_device > [ ,…n ]
[ WITH
  [ BLOCKSIZE = { blocksize | @blocksize_variable } ]
  [ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
  [ [ , ] DIFFERENTIAL ]
  [ [ , ] EXPIREDATE = { date | @date_var }]
  [ [ , ] PASSWORD = { password | @password_variable } ]
  [ [ , ] FORMAT | NOFORMAT ]
  [ [ , ] { INIT | NOINIT } ]
]
```

其中的参数说明如下：

(1) { database_name | @database_name_var }——指定了一个数据库，对该数据库进行完整的数据库备份或差异数据库备份。如果作为变量(@database_name_var)提供，则可

将该名称指定为字符串常量(@database_name_var = database name)或字符串数据类型(ntext 或 text 数据类型除外)的变量。

(2) < backup_device >——指定备份操作时要使用的逻辑或物理备份设备。可以是下列一种或多种形式:

① { logical_backup_device_name } | { @logical_backup_device_name_var }——备份设备的逻辑名称,数据库将备份到该设备中。

② { DISK | TAPE } = 'physical_backup_device_name' | @physical_backup_device_name_var——允许在指定的磁盘或磁带设备上创建备份。在执行 BACKUP 语句之前不必存在指定的物理设备。如果存在物理设备且 BACKUP 语句中没有指定 INIT 选项,则备份将追加到该设备。

(3) BLOCKSIZE = { blocksize | @blocksize_variable }——用字节数来指定物理块的大小。在 Windows NT 系统中,默认设置是设备的默认块大小。一般情况下,当 SQL Server 选择适合于设备的块大小时不需要此参数。

(4) DESCRIPTION = { 'text' | @text_variable }——指定描述备份集的自由格式文本。该字符串最长可以有 255 个字符。

(5) DIFFERENTIAL——指定数据库备份或文件备份应该与上一次完整备份后改变的数据库或文件部分保持一致。差异备份一般会比完整备份占用更少的空间。对于上一次完整备份时备份的全部单个日志,使用该选项可以不必再进行备份。

(6) EXPIREDATE = { date | @date_var }——指定备份集到期和允许被重写的日期。如果将该日期作为变量(@date_var)提供,则可以将该日期指定为字符串常量(@date_var = date)、字符串数据类型变量(ntext 或 text 数据类型除外)、smalldatetime 或者 datetime 变量,并且该日期必须符合已配置的系统 datetime 格式。

(7) PASSWORD = { password | @password_variable }——为备份集设置密码。PASSWORD 是一个字符串。如果为备份集定义了密码,则必须提供这个密码才能对该备份集执行任何还原操作。

(8) FORMAT——指定应将介质头写入用于此备份操作的所有卷。任何现有的介质头都被重写。FORMAT 选项使整个介质内容无效,即格式化备份设备。

(9) NOFORMAT——指定介质头不应写入所有用于该备份操作的卷中,并且不会格式化备份设备。除非指定了 INIT。

(10) INIT——表示如果备份集已经存在,新的备份集会覆盖旧的备份集。不会格式化备份设备。

(11) NOINIT——表示新的备份集会追加到旧的备份集的后面,不会覆盖。不会格式化备份设备。

注意:如果要备份特定的文件或文件组,则在 BACKUP DATABASE 语句中加入< file_or_filegroup >[,...n]参数即可;如果要进行事务日志备份,则使用 BACKUP LOG。详细内容请参考 Microsoft SQL Server 2008 的联机帮助。

【例 11-2】 将整个 teaching 数据库完整备份到磁盘上,并创建一个新的介质集。

```
BACKUP DATABASE teaching
TO DISK = 'G:\BACKUP\teaching.Bak'
```

```
    WITH FORMAT,
    NAME = 'teaching 的完整备份'
```

命令执行后，在对象资源管理器中展开 teaching 数据库，右击 teaching，在弹出的快捷菜单中选择“任务”→“备份”命令，弹出“备份数据库-teaching”窗口，就可以看到创建后的备份文件，如图 11-6 所示。

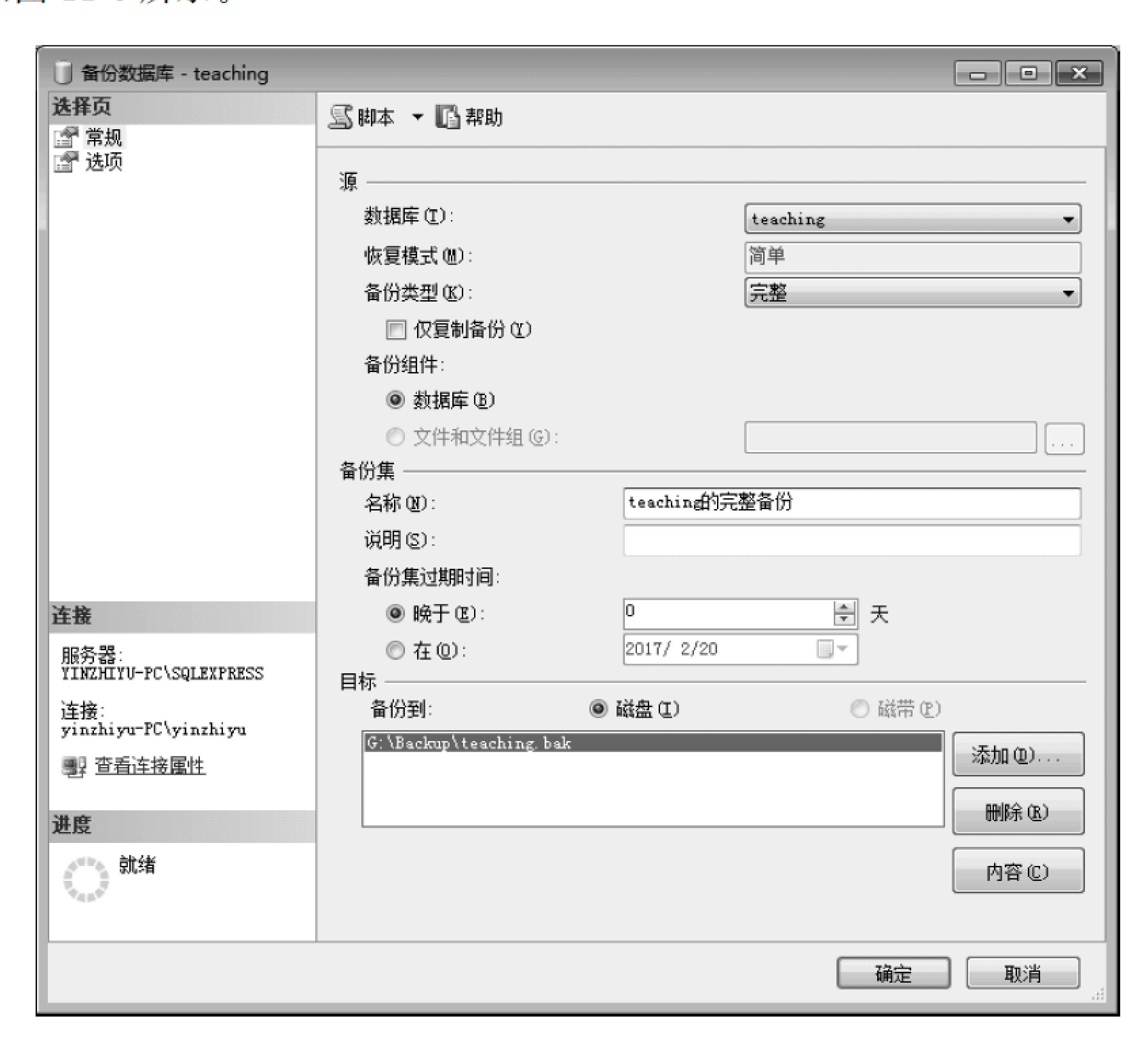


图 11-6　teaching 的完整数据库备份

在 teaching 数据库中，创建一个任意的新表，表名为 Table_1。

【例 11-3】 创建 teaching 的差异数据库备份。

```
BACKUP DATABASE teaching
TO DISK = 'G:\BACKUP\teaching 差异备份.Bak'
    WITH DIFFERENTIAL
```

命令执行后，可以以与上例相同的方法查看创建后的备份文件。

11.1.4　数据库还原操作

SQL Server 提供了数据库的两种还原过程：自动还原过程和手动还原过程。

1. 自动还原

自动还原是指 SQL Server 数据库在每次出现错误或关机重启之后都会自动运行带有容错功能的特性。SQL Server 用事务日志来完成这项任务，它读取每个数据库事务日志的活动部分，并且检查所有自最新的检查点以来发生的事务。检查点就是最近一次从内存中把数据变化永久写入到数据库中的那个时间点，标识所有已经提交的事务，把它们重新应用

于数据库，然后标识所有未提交的事务并回滚，这样保证删除所有未完全写入数据库的未提交事务。这个过程保证了每个数据库逻辑上的一致性。

SQL Server 最先还原 master 数据库，接着还原 model 数据库和 msdb 数据库，然后还原每一个用户数据库，最后清除并启动 tempdb 数据库，结束还原过程。

2. 手动还原

手动还原数据库需要指定数据库还原工作的应用程序和接下来的按照创建顺序排列的事务日志的应用程序。完成这些之后，数据库就会处于和事务日志最后一次备份时一致的状态。

如果使用完全数据库备份来还原，SQL Server 重新创建这些数据库文件和所有的数据库对象；如果使用差异数据库备份来还原，则可以还原最近的差异数据库备份。

在 Microsoft SQL Server 2008 中，数据库还原操作有两种方式：在 SSMS 中使用界面还原数据库和使用 T-SQL 语句还原数据库。

1）在 SSMS 中使用界面还原数据库

【例 11-4】 在 SSMS 的“对象资源管理器”中利用 teaching 的完整数据库备份还原 teaching 数据库，操作步骤如下：

（1）在对象资源管理器中展开 teaching 数据库。

（2）右击 teaching，在弹出的快捷菜单中选择“任务”→“还原”→“数据库”命令，弹出“还原数据库-teaching”窗口。

（3）选择要还原的目标数据库为 teaching，选择用于还原的备份集为 teaching 的完整备份，如图 11-7 所示。选择左侧窗格的“选项”选项。

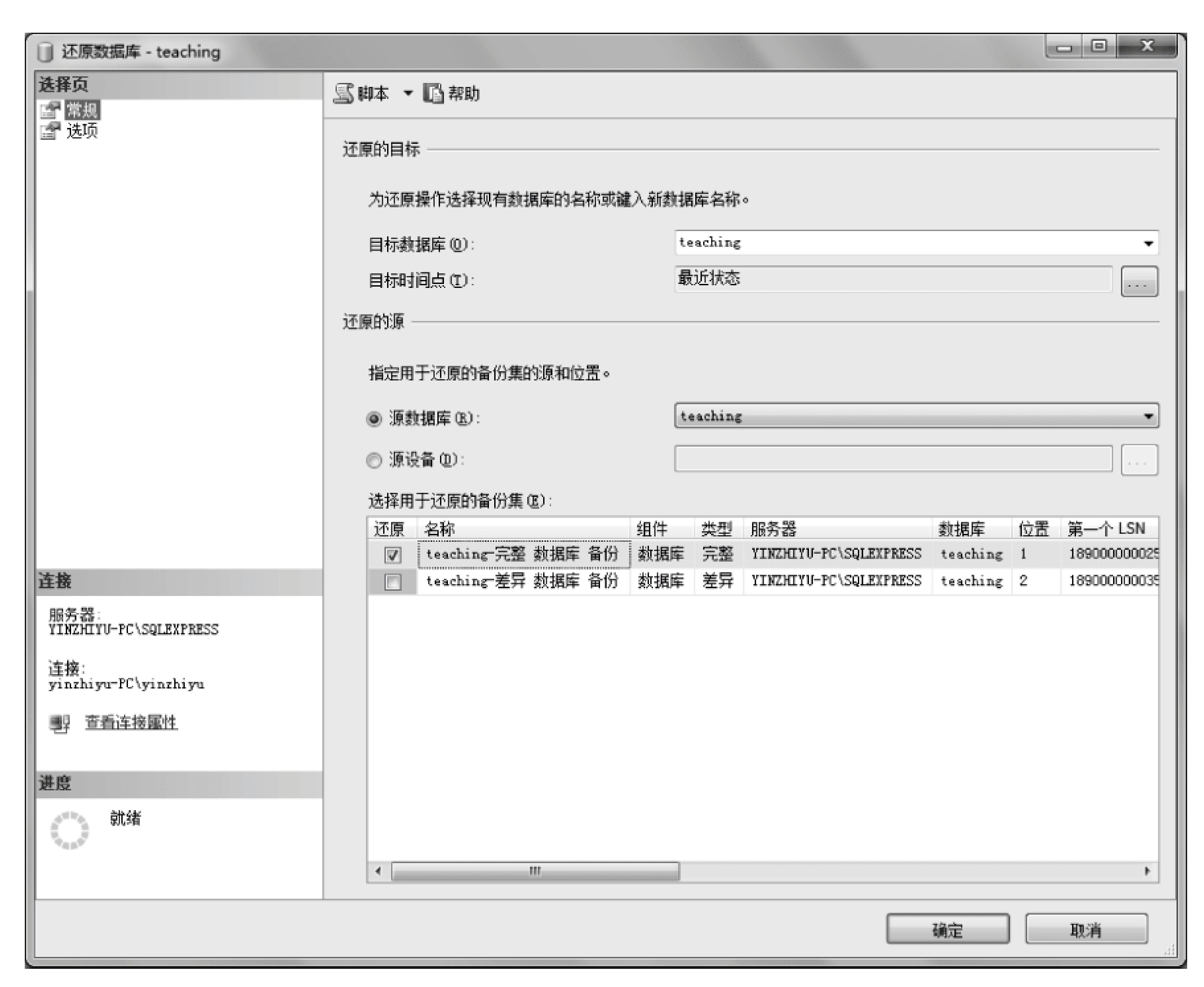


图 11-7　还原数据库-teaching

（4）在“选项”选择页中选择“覆盖现有数据库”复选框；在“将数据库文件还原为”中查看或修改“原始文件名”和“还原为”中的文件名，此处都为 teaching 的相应文件；在“恢复状态”下拉列表框中选择需要的选项，此处为默认的第一项，如图 11-8 所示。单击“确定”按钮，数据库还原操作完成。

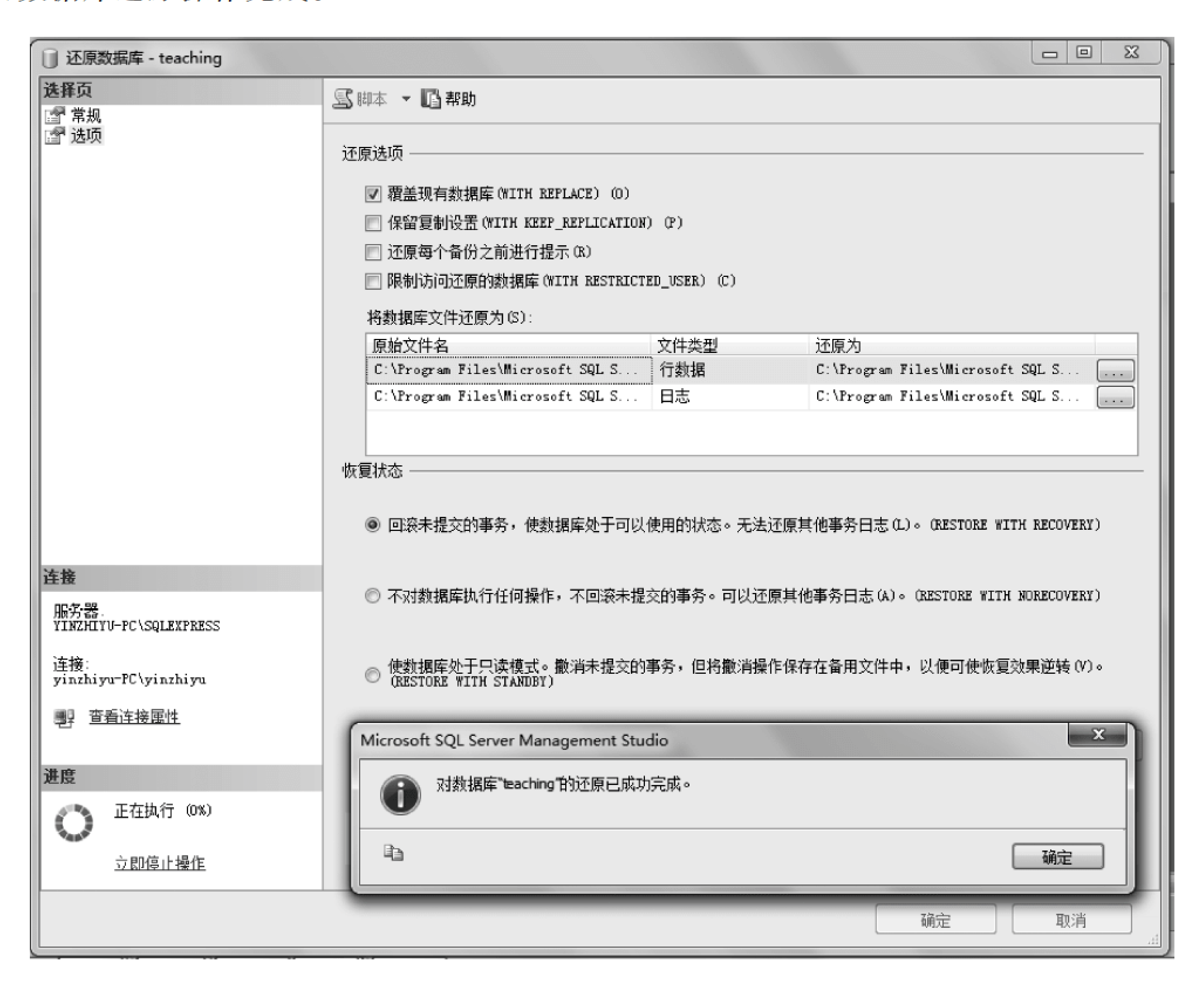


图 11-8　“选项”选择页

打开 teaching 数据库，可以看到其中的数据进行了还原。看不到其中的 Table_1 表，因为只进行了完整数据库备份的还原。

【例 11-5】　在“对象资源管理器”中利用 teaching 的差异数据库备份还原 teaching，操作步骤和还原完整数据库备份基本相同。

在“还原数据库-teaching”窗口，选择用于还原的备份集为 teaching 的差异备份，“teaching 的完整备份”会自动被选中，因为在还原差异备份之前，必须先还原其基准备份，如图 11-9 所示，两个备份都选中。

还原操作完成后，打开 teaching，可以看到完整备份时的数据，也可以看到其中的 Table_1 表，因为还原了完整数据库备份后的差异数据库备份。

2）使用 T-SQL 语句还原数据库

使用 T-SQL 语句还原数据库的基本语法格式如下：

```
RESTORE DATABASE { database_name | @database_name_var }
[ FROM <backup_device> [ ,...n ] ]
[ WITH
```

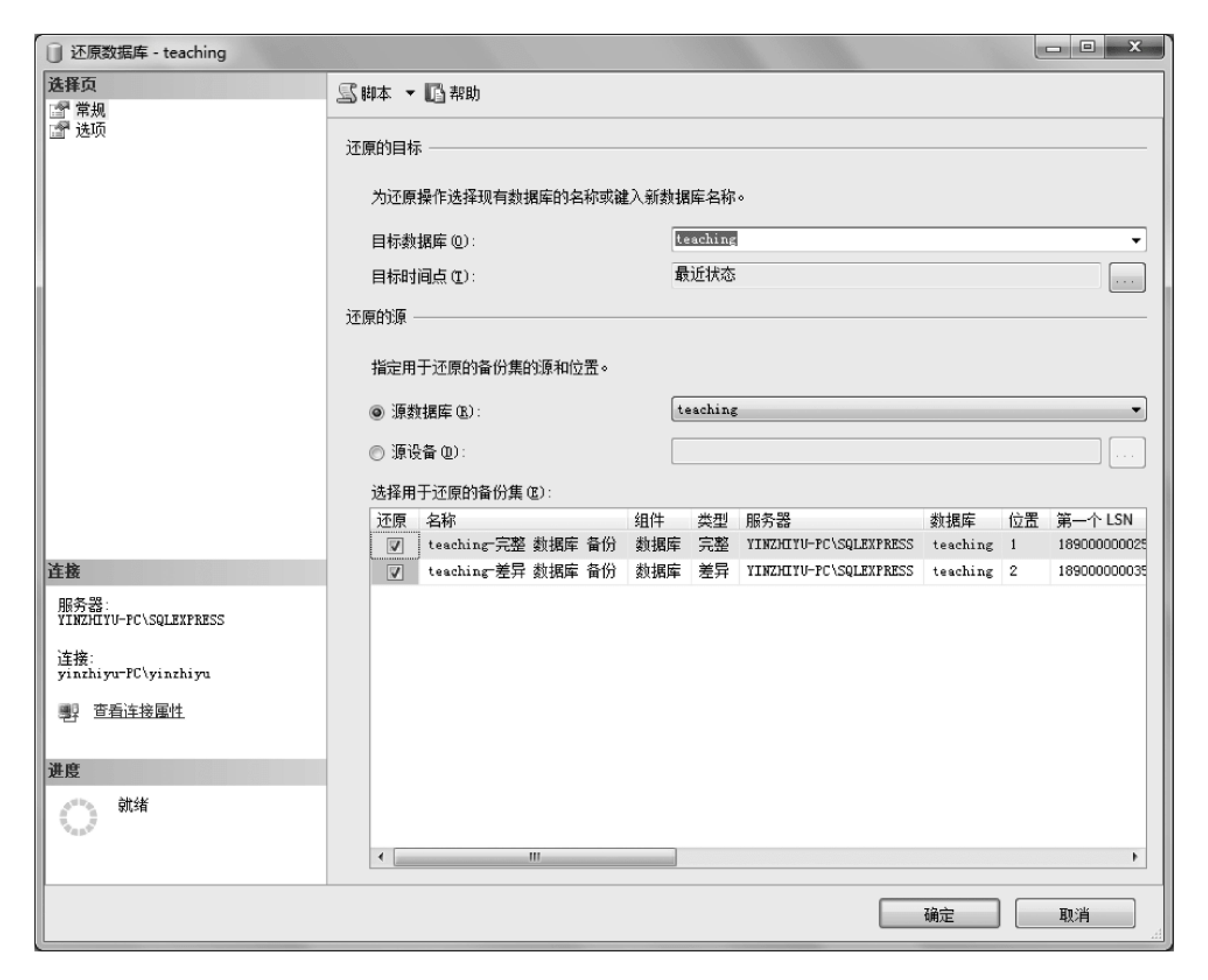


图 11-9　还原 teaching 的差异数据库备份

```
    [ [ , ] FILE = { backup_set_file_number | @backup_set_file_number } ]
    [ [ , ] KEEP_REPLICATION ]
    [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
    [ [ , ] MEDIAPASSWORD = { mediapassword | @mediapassword_variable } ]
  [ [ , ] MOVE 'logical_file_name_in_backup' TO 'operating_system_file_name' ]
[ ,...n ]
[ [ , ] PASSWORD = { password | @password_variable } ]
[ [ , ] { RECOVERY | NORECOVERY | STANDBY =
          {standby_file_name | @standby_file_name_var } } ]
[ [ , ] REPLACE ]
]
```

其中大部分参数在介绍备份数据时已经介绍过了，下面对一些没有介绍过的参数进行说明：

(1) KEEP_REPLICATION——将复制设置为与日志传送一同使用。设置该参数后，在备用服务器上还原数据库时，可防止删除复制设置。

(2) MOVE——将逻辑名指定的数据文件或日志文件还原到所指定的位置。

(3) RECOVERY——回滚未提交的事务，使数据库处于可以使用状态。无法还原其他事务日志。

(4) NORECOVERY——不对数据库执行任何操作，不回滚未提交的事务。可以还原其他事务日志。

(5) STANDBY——使数据库处于只读模式。撤销未提交的事务,但将撤销操作保存在备用文件中,以便可以恢复效果逆转。

(6) standby_file_name | @standby_file_name_var——指定一个允许撤销恢复效果的备用文件或变量。

(7) REPLACE——会覆盖所有现有数据库以及相关文件,包括已存在的同名的其他数据库或文件。

【例 11-6】 将 teaching 数据库的完整数据库备份进行还原。

```
RESTORE DATABASE teaching
FROM DISK = 'G:\BACKUP\teaching.Bak'
WITH REPLACE, NORECOVERY
```

【例 11-7】 将 teaching 数据库的差异数据库备份进行还原。

```
RESTORE DATABASE teaching
FROM DISK = 'G:\BACKUP\teaching差异备份.Bak'
WITH RECOVERY
```

11.2 数据库分离与附加

SQL Server 2008 允许分离数据库的数据和事务日志文件,然后将其重新附加到同一台或另一台服务器上。分离数据库将从 SQL Server 删除数据库,但是保证在组成该数据库的数据和事务日志文件中的数据库完好无损。然后这些数据和事务日志文件可以用来将数据库附加到任何 SQL Server 实例上,这使数据库的使用状态与它分离时的状态完全相同。

例如,如果数据库系统安装在系统盘(比如 C 盘),由于 C 盘容易受病毒侵害,所以也许希望数据存放在非系统盘(比如 D 盘),要做到这点很简单——不需要重装数据库,只要把数据库"分离",然后将相关文件移动到 D 盘的某个目录,接着"附加"数据库即可。

11.2.1 分离数据库

在 Microsoft SQL Server 2008 中,数据库分离操作有两种方式:在 SSMS 中使用界面分离数据库和使用 T-SQL 语句分离数据库。

1. 在 SSMS 中使用界面分离数据库

在 SSMS"对象资源管理器"中分离数据库的操作步骤如下:

(1) 在"对象资源管理器"中展开要分离的数据库。

(2) 右击数据库名称,在弹出的快捷菜单中选择"任务"→"分离"命令,如图 11-10 所示。

(3) 弹出"分离数据库"对话框,如图 11-11 所示。单击"确定"按钮即可完成数据库的分离。

打开"对象资源管理器",被分离的数据库就不存在了。但是,在存储此数据库的物理位置(即某磁盘目录下),其数据文件和日志文件仍然存在,可以任意复制。



图 11-10 在“对象资源管理器”中分离数据库

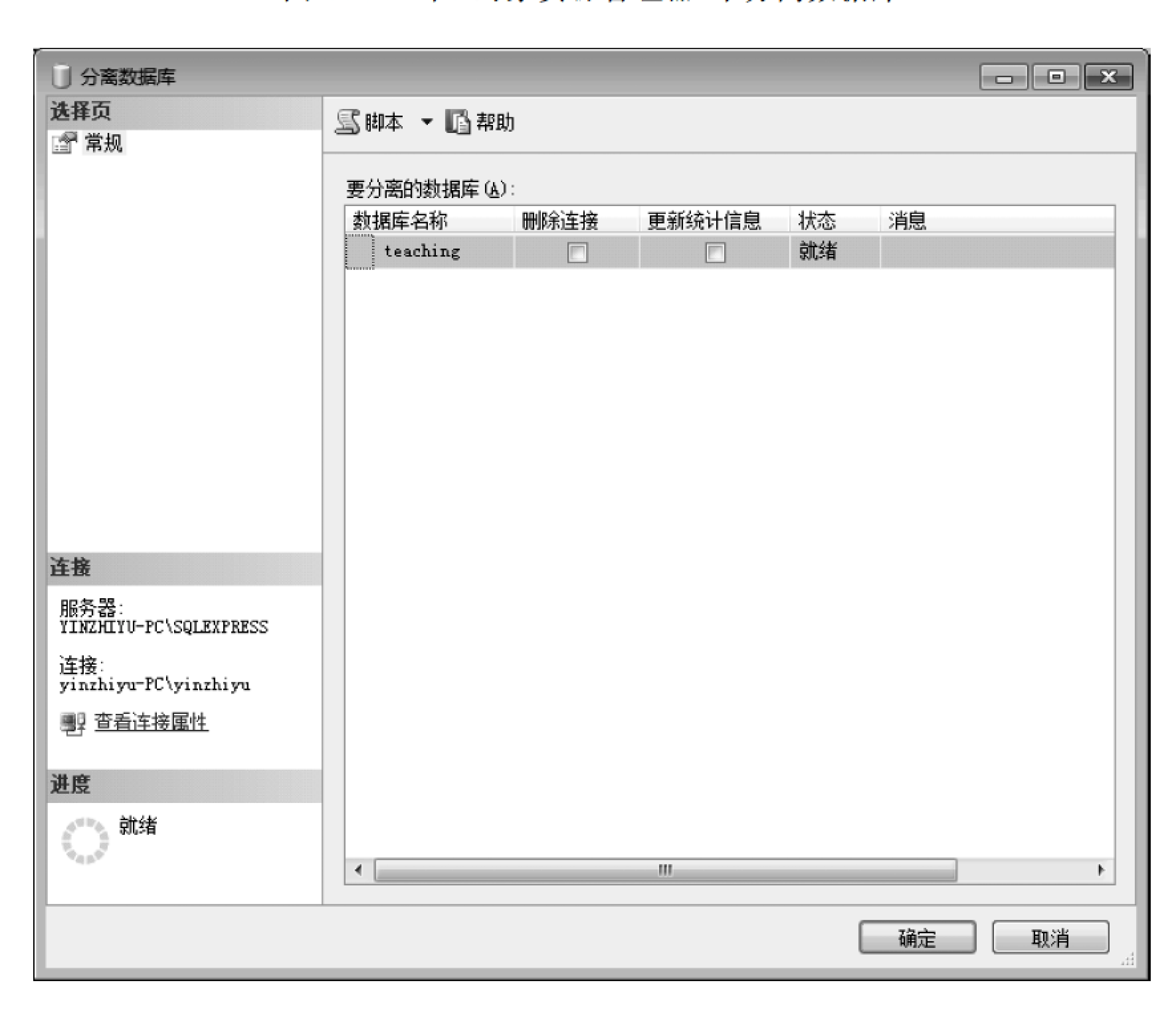


图 11-11 “分离数据库”对话框

注意：只有"使用本数据库的连接"数为 0 时，该数据库才能分离。所以分离数据库时尽量断开所有对要分离数据库操作的连接，如果还有连接数据库的程序，则会出现分离数据库失败的提示对话框，如图 11-12 所示。可以在图 11-11 中选中"删除连接"复选框，从服务器强制断开现有的连接。

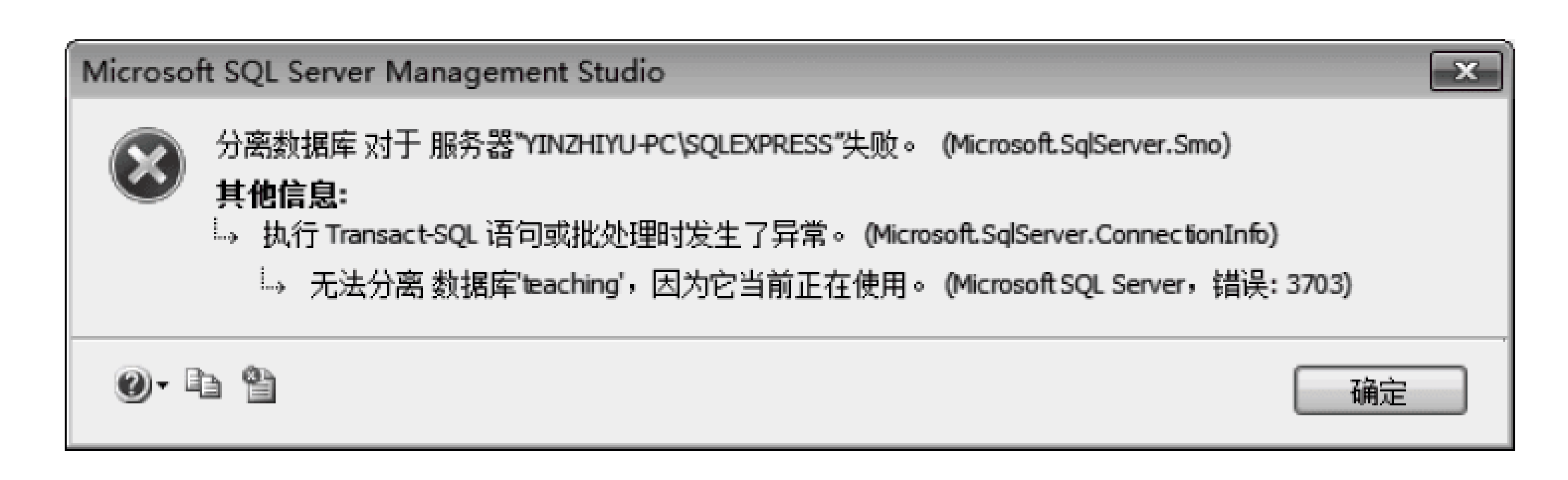


图 11-12　分离数据库失败

2. 使用 T-SQL 语句分离数据库

可以使用系统存储过程 sp_detach_db 分离该数据库。sp_detach_db 存储过程从服务器分离数据库，并可以选择在分离前在所有的表上运行 UPDATE STATISTICS。

其语法格式如下：

```
sp_detach_db [ @dbname = ] 'dbname'
    [ , [ @skipchecks = ] 'skipchecks' ]
```

参数说明如下：

(1) [@dbname =] 'dbname'——要分离的数据库名称。@dbname 的数据类型为 sysname，默认值为 NULL。

(2) [@skipchecks =] 'skipchecks'——@skipchecks 的数据类型为 nvarchar(10)，默认值为 NULL。如果为 true，则跳过 UPDATE STATISTICS；如果为 false，则运行 UPDATE STATISTICS。对于要移动到只读介质上的数据库，此选项很有用。

【例 11-8】 分离 teaching 数据库，并将 skipchecks 设为 true。

```
EXEC sp_detach_db 'teaching', 'true'
```

11.2.2　附加数据库

与分离对应的是附加数据库操作。附加数据库可以很方便地在 SQL Server 2008 服务器之间利用分离后的数据文件和日志文件组织成新的数据库。数据库的附加好比是将衣服(数据库)重新挂上衣架(SQL Server 2008 服务器)。

在 Microsoft SQL Server 2008 中，数据库附加操作有两种方式：在 SSMS 中使用界面附加数据库和使用 T-SQL 语句附加数据库。

1. 在 SSMS 中使用界面附加数据库

在 SSMS 的"对象资源管理器"中附加数据库的操作步骤如下：

(1) 在"对象资源管理器"中右击"数据库"，选择"附加"命令，如图 11-13 所示。

(2) 在弹出的"附加数据库"对话框中，单击"添加"按钮，如图 11-14 所示。

(3) 在弹出的"定位数据库文件"对话框中，选择要附加的磁盘上的数据库文件，再单击



图 11-13　在对象资源管理器中附加数据库

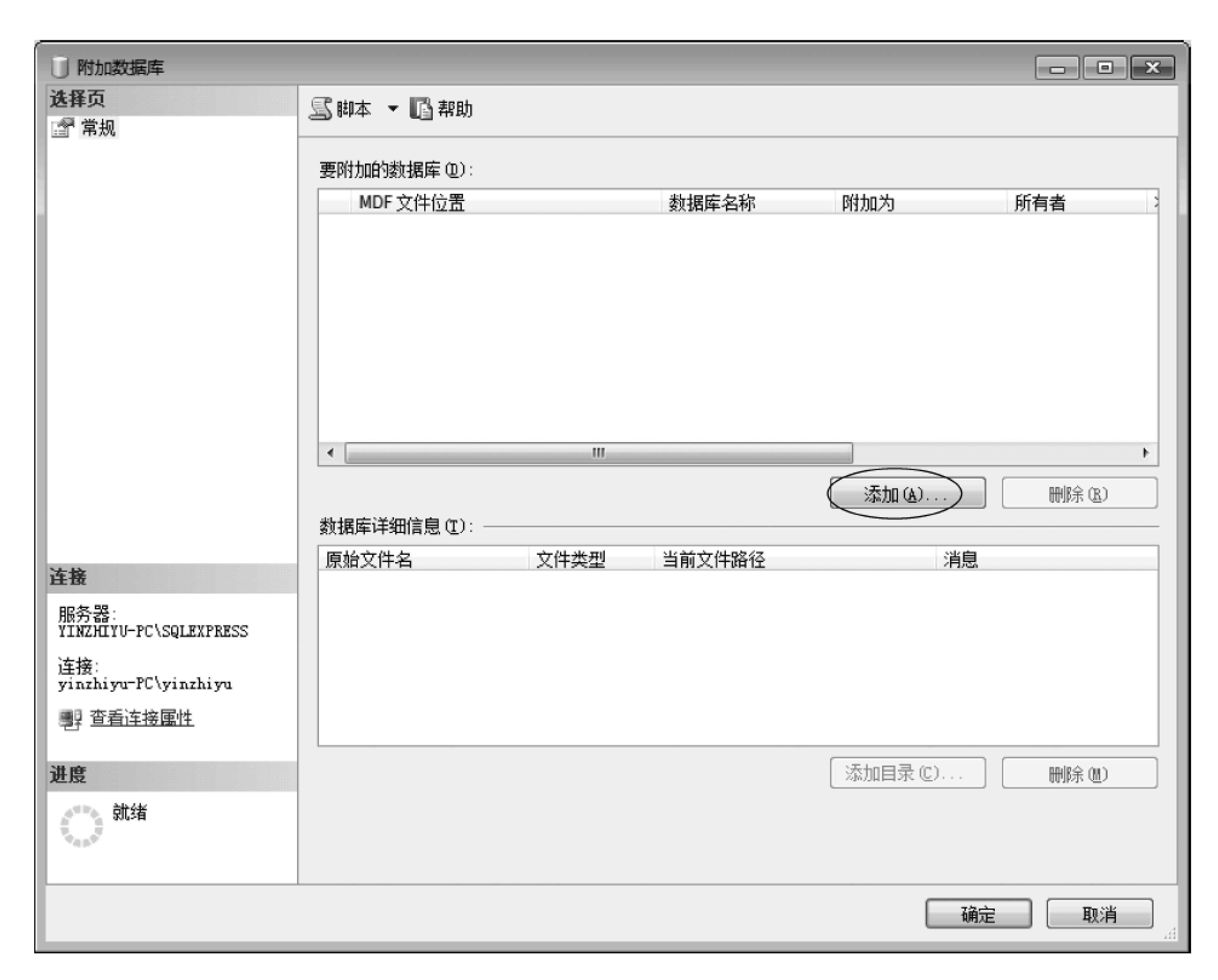


图 11-14　“附加数据库”对话框

“确定”按钮，如图 11-15 所示。

(4) 进入如图 11-16 所示的窗口后，就可以看到添加进来的数据库的数据文件和日志文件，单击“确定”按钮，完成数据库的附加。

2. 使用 T-SQL 语句附加数据库

可以使用系统存储过程 sp_attach_db 将数据库附加到当前服务器或使用系统存储过程 sp_attach_single_file_db 将只有一个数据文件的数据库附加到当前服务器。

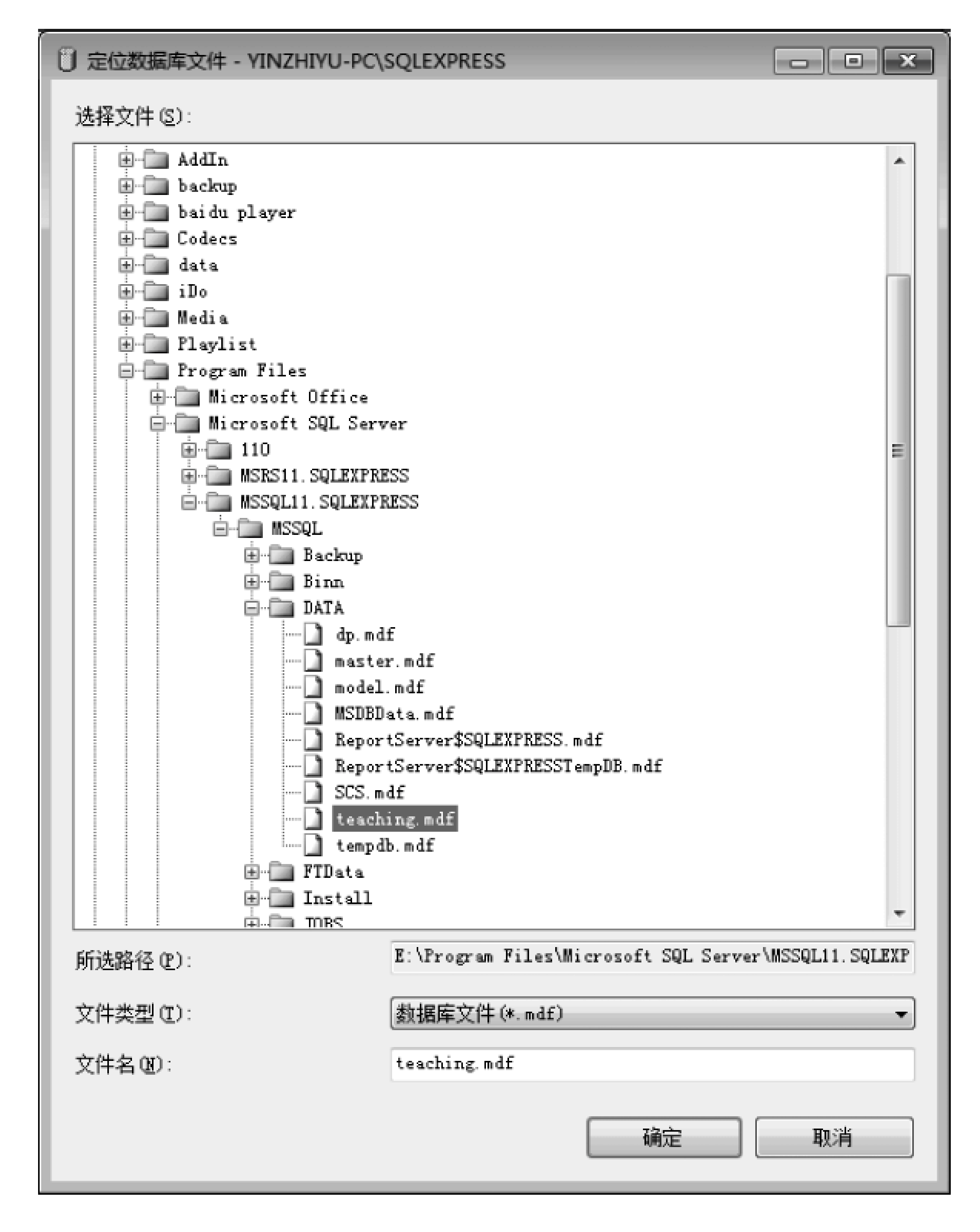


图 11-15　附加的磁盘上的数据库文件

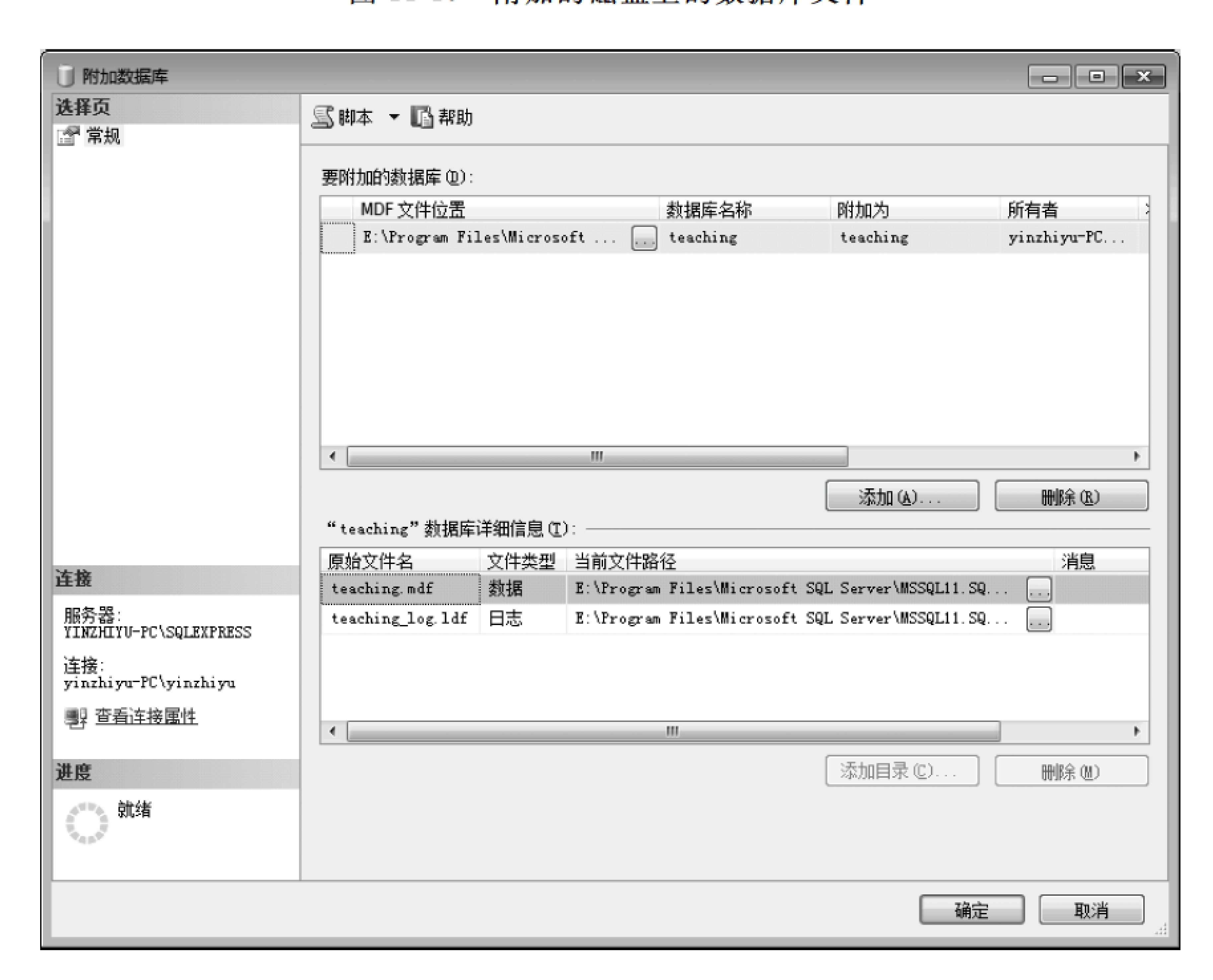


图 11-16　附加数据库完成

(1) 系统存储过程 sp_attach_db 附加数据库的语法格式：

```
sp_attach_db [ @dbname = ] 'dbname', [ @filename1 = ] 'filename_n'[ ,...16 ]
```

参数说明如下：

① [@dbname =] 'dbname'——要附加到服务器的数据库的名称。该名称必须是唯一的。dbname 的数据类型为 sysname，默认值为 NULL。

② [@filename1 =] 'filename_n'——数据库文件的物理名称，包括路径。filename_n 的数据类型为 nvarchar(260)，默认值为 NULL。最多可以指定 16 个文件名。参数名称以@filename1 开始，递增到@filename16。文件名列表至少必须包括主文件，主文件包含指向数据库中其他文件的系统表。该列表还必须包括数据库分离后所有被移动的文件。

【例 11-9】 附加 teaching 数据库到当前服务器。

```
EXEC sp_attach_db @dbname = 'teaching',
@filename1 = 'E:\Program Files\Microsoft SQL Server \MSSQL11.SQLEXPRESS\ MSSQL\DATA\
teaching.mdf',
@filename2 = 'E:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\ MSSQL\DATA\
teaching_log.ldf'
```

(2) 系统存储过程 sp_attach_single_file_db 附加只有一个数据文件的数据库的语法格式：

```
sp_attach_single_file_db [ @dbname = ] 'dbname', [ @physname = ] 'physical_name'
```

其中，[@physname =] 'physical_name'为数据库文件的物理名称，包括路径。physical_name 的数据类型为 nvarchar(260)，默认值为 NULL。

【例 11-10】 附加 teaching 数据库到当前服务器。

```
EXEC sp_attach_single_file_db @dbname = 'teaching',@physname = 'E:\Program Files\ Microsoft
SQL Server \MSSQL11.SQLEXPRESS\ MSSQL\DATA\teaching.mdf'
```

分离和附加数据库的操作可以将数据库从一台计算机移到另一台计算机，而不必重新创建数据库，当附加到数据库上时，必须指定主数据文件的名称和物理位置。主文件包含查找由数据库组成的其他文件所需的信息。如果存储的文件位置发生了改变，就需要手工指定次要数据文件和日志文件的存储位置。

11.3 数据库快照

数据库快照(database snapshot)是 SQL Server 数据库(源数据库)的只读静态视图，其本质类似于源数据库的照片。自创建快照的那一刻起，数据库快照在事务上与源数据库完全相同。数据库快照始终与其源数据库位于同一服务器实例上，当源数据库更新时，数据库快照也将更新。因此，数据库快照存在的时间越长，就越有可能用完其可用磁盘空间。

一个源数据库中可以存在多个快照，在数据库所有者显式删除每个数据库快照之前，该快照将一直保留，但如果源数据库因某种原因而不可用，则它的所有数据库快照也将不可用。

11.3.1 数据库快照功能概述

数据库快照在数据页级运行。在第一次修改源数据库页之前,先将原始页从源数据库复制到快照。快照将存储原始页,保留它们在创建快照时的数据记录,对要进行第一次修改的每一页重复此过程。对于用户而言,数据库快照似乎始终保持不变,因为对数据库快照的读操作始终访问原始数据页,而与页驻留的位置无关。

为了存储复制的原始页,快照使用一个或多个"稀疏文件(sparse file)"。稀疏文件是NTFS文件系统提供的文件,需要的磁盘空间要比其他文件格式少很多,用于存储复制到数据库快照的页面。

最初,稀疏文件实质上是空文件,不包含用户数据并且未被分配存储用户数据的磁盘空间。随着数据写入数据库快照,NTFS会将磁盘空间逐渐分配给相应的稀疏文件,随着源数据库中更新的页越来越多,文件的大小也不断增长。图11-17说明了两种相对的更新模式对快照大小的影响,更新模式A反映的是在快照使用期限内仅有30%的原始页更新的环境,更新模式B反映的是在快照使用期限内有80%的原始页更新的环境。

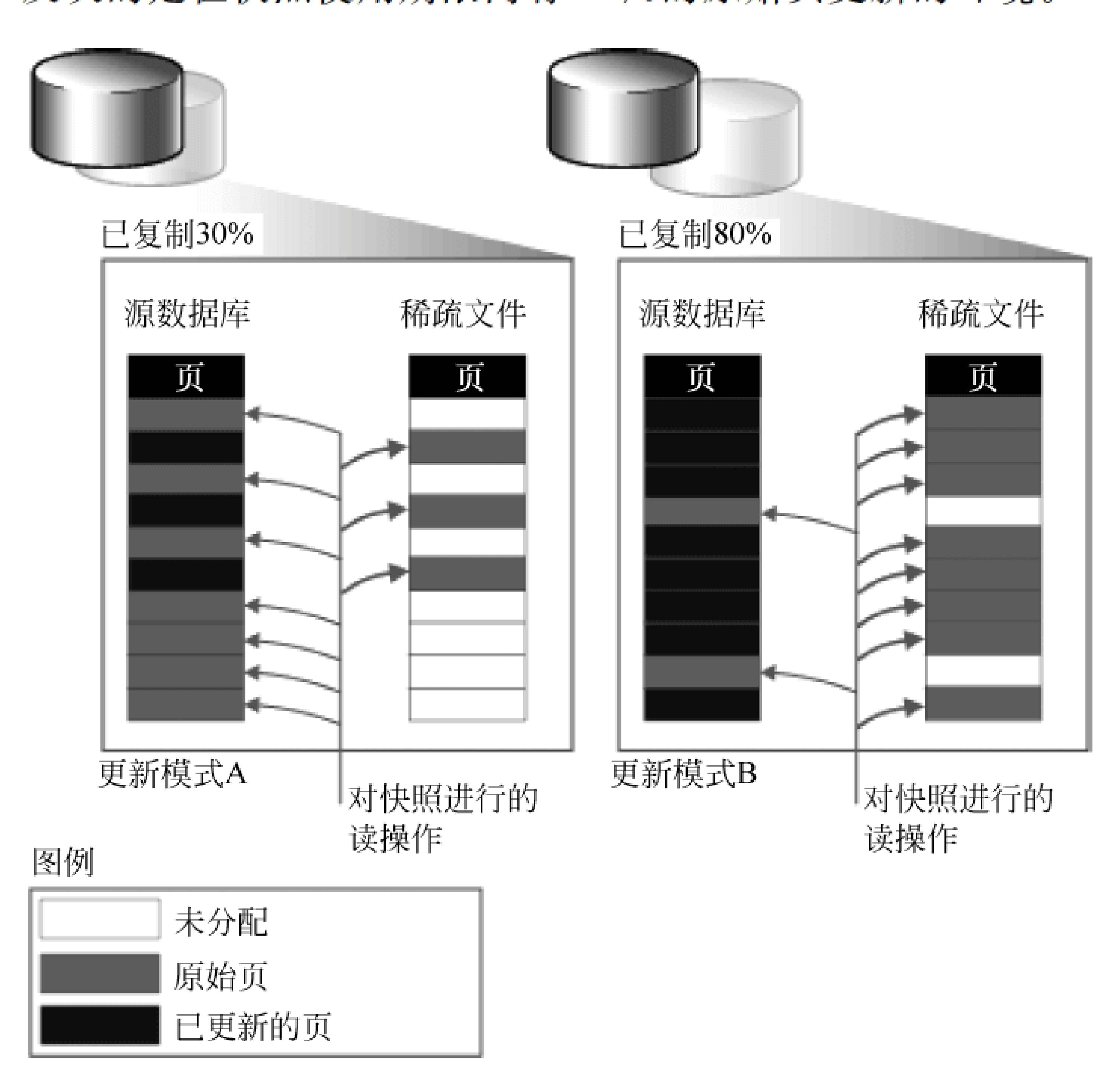


图 11-17 数据库快照的更新模式

11.3.2 数据库快照的优点

数据库快照具有以下几个优点:

1. 快照可用于报告目的

客户端可以查询数据库快照,这对于基于创建快照时的数据编写报表是很有用的。

2. 维护历史数据以生成报表

快照可以从特定时点扩展用户对数据的访问权限。例如,可以在给定时间段(例如,财

务季度)要结束的时候创建数据库快照以便日后制作报表,然后可以在快照上运行当时创建的报表。如果磁盘空间允许,还可以维护任意多个不同期间要结束时的快照,以便能够对这些时间段的结果进行查询。

3. 使用为了实现可用性目标而维护的镜像数据库来减轻报表负载

使用带有数据库镜像的数据库快照,使用户能够访问镜像服务器上的数据以生成报表。而且,在镜像数据库上运行查询可以释放主体数据库上的资源。

4. 使数据免受管理失误所带来的影响

如果源数据库上出现用户错误,则可将源数据库恢复到创建给定数据库快照时的状态,丢失的数据仅限于创建该快照后数据库中发生更新的数据。

例如,在进行重大更新(比如大容量更新或架构更改)前,对数据库创建数据库快照以保护数据。为了更好地保护数据,可以创建时间跨度足以识别和处理大多数用户错误的一系列数据库快照。例如,根据磁盘资源,可以每 24 小时创建 6～12 个滚动快照。每创建一个新的快照,就删除最早的快照。

若要从用户错误中恢复,可以将数据库恢复到在错误发生的前一时刻的快照。为此目的进行的恢复很可能比从备份还原快得多;但是,此后无法对数据进行回滚操作。或者,也可以利用快照中的信息,手动重新创建删除的表或其他丢失的数据。例如,可以将快照中的数据大容量复制到数据库中,然后手动将数据合并回数据库中。

5. 管理测试数据库

在测试环境中,当每一轮测试开始时针对要包含相同数据的数据库重复运行测试协议将十分有用。在运行第一轮测试前,应用程序开发人员或测试人员可以在测试数据库中创建数据库快照。每次运行测试之后,数据库都可以通过恢复数据库快照快速返回到它以前的状态。

注意:使用数据库快照的原因,决定了数据库需要多少个并发快照、多久创建一次新快照以及将其保留多久。无法对脱机或损坏的数据库进行恢复,因此,使用数据库快照还原数据库不能代替备份和还原策略,严格按计划执行备份仍然至关重要。

11.3.3 数据库快照的先决条件和限制

1. 先决条件

可以使用任何恢复模式的源数据库必须满足以下先决条件:

(1) 服务器实例必须在支持数据库快照的 SQL Server 版本上运行,SQL Server 2005 以上的企业版才支持数据库快照功能。

(2) 源数据库必须处于联机状态,除非该数据库在数据库镜像会话中是镜像数据库。若要在镜像数据库中创建数据库快照,数据库必须处于 SYNCHRONIZED 镜像状态。

(3) 不能将源数据库配置为可缩放共享数据库。

2. 源数据库的限制

只要存在数据库快照,快照的源数据库就存在以下限制:

(1) 不能对源数据库进行删除、分离或还原。

(2) 由于每次更新页时都会对快照执行“写入时复制”操作,导致源数据库上的 I/O 增加,所以源数据库的性能会受到影响。

(3) 不能从源数据库或任何快照中删除文件。

3. 数据库快照的限制

数据库快照存在以下限制：

(1) 数据库快照必须与源数据库在相同的服务器实例上创建和保留。

(2) 始终对整个数据库制作数据库快照。

(3) 数据库快照依赖于源数据库，但不是冗余存储。它们无法防止磁盘错误或其他类型的损坏。因此，使用数据库快照恢复数据库不能代替备份和还原策略。当将源数据库中更新的页强制压入快照时，如果快照用尽磁盘空间或者遇到其他错误，则该快照将成为可疑快照并且必须将其删除。

(4) 快照为只读的。

(5) 不能备份或还原数据库快照，不能附加或分离数据库快照。

(6) 不能在 FAT32 文件系统或 RAW 分区上创建数据库快照，数据库快照所用的稀疏文件由 NTFS 文件系统提供。

(7) 数据库快照将继承快照创建时其源数据库的安全约束。由于快照是只读的，因此无法更改继承的权限，对源数据库的更改权限将不反映在现有快照中。

(8) 快照始终反映创建该快照时的文件组状态：联机文件组将保持联机状态，脱机文件组将保持脱机状态。

4. 磁盘空间要求

(1) 数据库快照占用磁盘空间。如果数据库快照用尽了磁盘空间，将被标记为可疑，必须将其删除。但是，源数据库不会受到影响，对其执行的操作仍能继续正常进行。通常情况下，快照只会保留一段有限的时间，因此其大小不是主要问题。

(2) 保留快照的时间越长，越有可能将可用空间用完。稀疏文件最大只能增长到创建快照时相应的源数据库文件的大小。

11.3.4 SQL Server 2008 的数据库快照操作

1. 建立数据库快照

SQL Server 不支持在 SSMS 中使用界面建立数据库快照，只能通过 T-SQL 语句创建。创建数据库快照的 T-SQL 语句语法格式如下：

```
CREATE DATABASE database_snapshot_name
  ON
  ( NAME = logical_file_name,
    FILENAME = 'os_file_name'
  ) [ ,...n]
  AS SNAPSHOT OF source_database_name
```

参数说明如下：

① database_snapshot_name——要创建的数据库快照名称。因为一个数据库可以创建多个快照，所以为了区分，可以用源数据库名加创建快照的日期和时间为其命名。

② NAME = logical_file_name——指定源数据库的数据文件的逻辑名。

③ FILENAME = 'os_file_name'——指定数据库快照稀疏文件的物理路径和文件名。

④ source_database_name——指定要创建的数据库快照的源数据库逻辑名。

【例 11-11】 创建 teaching 数据库的快照。

```
CREATE DATABASE teaching_20170321_11am ON
(Name = 'teaching',FileName = 'D:\data\teaching_20170321_11am.snap')
AS SNAPSHOT OF teaching
```

数据库快照创建成功后，可以像源数据库一样在 SSMS 中查看，如图 11-18 所示。

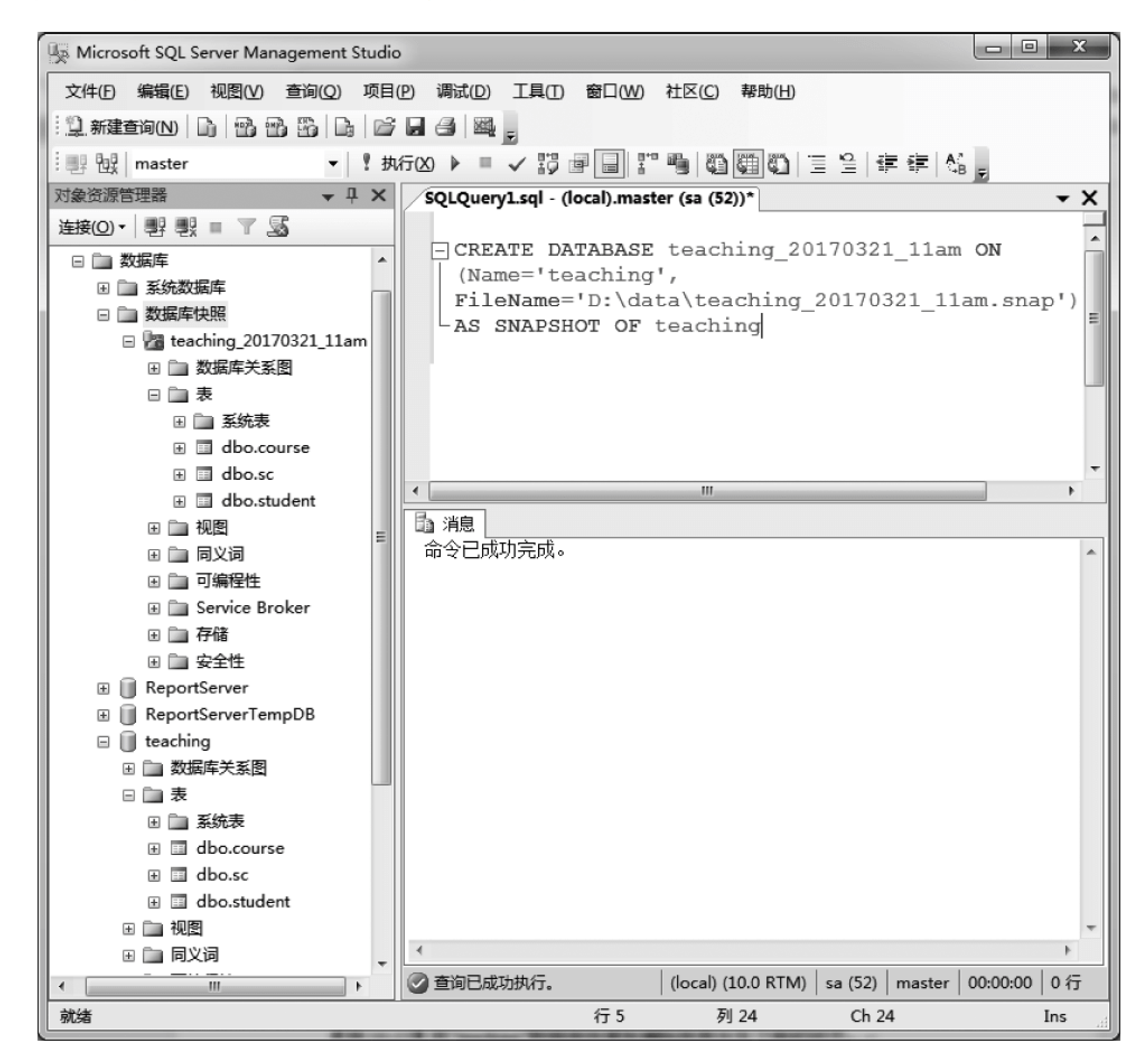


图 11-18 创建 teaching 数据库的快照

因为数据库快照是只读的，所以用户只能查询其中的数据，不能更新。

【例 11-12】 从 teaching 数据库的学生表中将所有女生的信息删除，然后查询所有学生的基本信息，再查询 teaching_20170321_11am 数据库快照中所有学生的基本信息。操作结果如图 11-19 所示。

2. 通过快照恢复数据库

当需要将源数据库恢复到快照时的状态时，可以使用 RESTORE 命令通过数据库快照恢复数据库，语法格式如下：

```
RESTORE DATABASE source_database_name
FROM DATABASE_SNAPSHOT = 'database_snapshot_name'
```

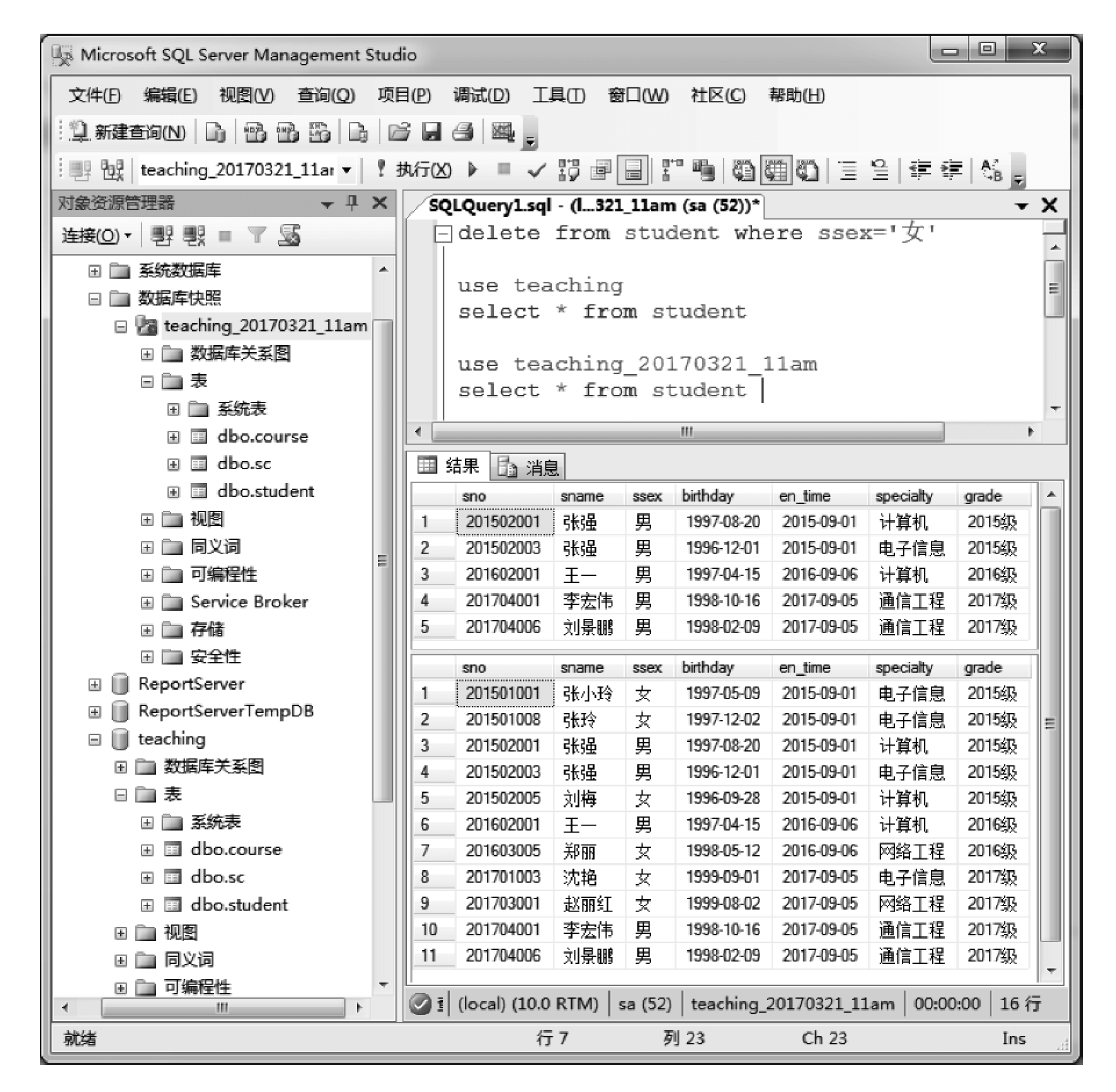


图 11-19 源数据库和快照中数据的查询结果对比

【例 11-13】 应用 teaching_20170321_11am 数据库快照将 teaching 数据库恢复到删除所有女生信息之前的状态。

```
RESTORE DATABASE teaching
FROM DATABASE_SNAPSHOT = 'teaching_20170321_11am'
```

操作结果如图 11-20 所示。

3. 删除数据库快照

当同一个数据库存在多个数据库快照时,不能通过任何一个快照恢复数据库,必须只留下用于恢复数据库的快照,其他的全部删除。删除数据库快照的命令与删除数据库的命令相同,语法格式为:

```
DROP DATABASE database_snapshot_name
```

【例 11-14】 将数据库快照 teaching_20170321_11am 删除。

```
DROP DATABASE teaching_20170321_11am
```

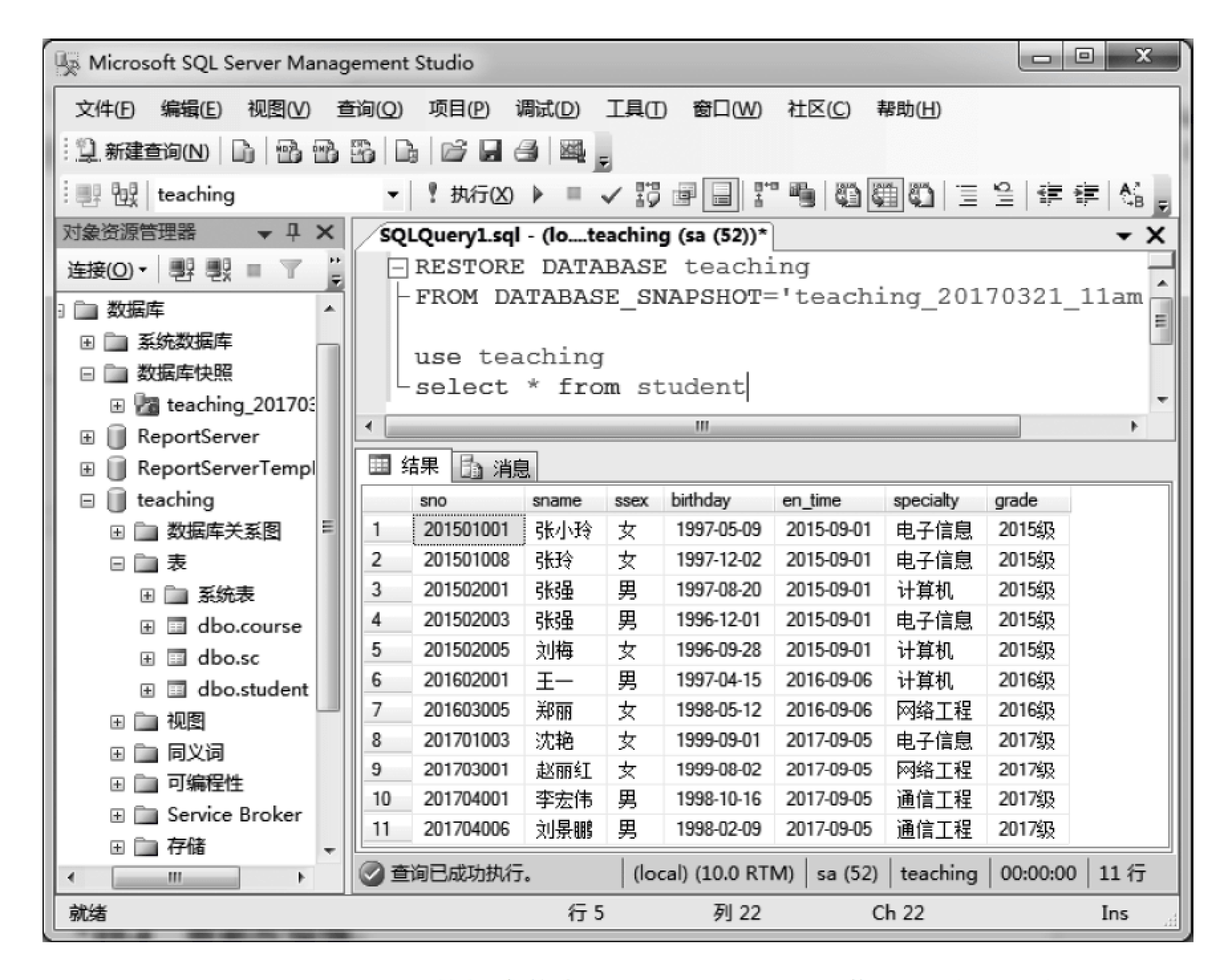


图 11-20 teaching 数据库恢复到了删除所有女生信息之前的状态

11.4 数据库镜像

数据库镜像(database mirroring)是一种提高 SQL Server 数据库可用性的解决方案。镜像基于每个数据库实现,并且只适用于使用完整恢复模式的数据库。

11.4.1 数据库镜像概述

数据库镜像是 SQL Server 2005 以后版本具有的功能,它在不同的数据库引擎服务器实例维护一个数据库的两个副本,将事务日志记录直接从一台服务器传输到另一台服务器,并且能够在出现故障时快速转移到备用服务器。可以编写客户端程序自动重定向连接信息,这样一旦出现故障转移就可以自动连接到备用服务器和数据库。

1. 工作方式

数据库的两个副本必须驻留在不同的 SQL Server 数据库引擎服务器实例上,这些服务器实例驻留在不同位置,甚至是不同地域的计算机上。启动数据库上的数据库镜像操作时,在这些服务器实例之间形成一种关系,称为“数据库镜像会话”。

其中一个服务器(主体服务器)实例使数据库服务于客户端,另一个服务器实例则根据镜像会话的配置和状态,充当热备用或温备用服务器(镜像服务器)。同步数据库镜像会话时,数据库镜像提供热备用服务器,可支持在已提交事务不丢失数据的情况下进行快速故障转移;未同步会话时,镜像服务器通常用作温备用服务器(可能造成数据丢失)。

在“数据库镜像会话”中，主体服务器和镜像服务器作为“伙伴”进行通信和协作，并双方互补。两个伙伴在会话中扮演互补的角色：“主体角色”和“镜像角色”。在任何给定的时间，每个伙伴都拥有其当前角色，一个扮演主体角色，另一个扮演镜像角色。拥有主体角色的伙伴称为“主体服务器”，其数据库为当前主体数据库；拥有镜像角色的伙伴称为“镜像服务器”，其数据库为当前镜像数据库。

数据库镜像会尽快将对主体数据库执行的每项插入、更新和删除等操作“重做”到镜像数据库中。重做通过将活动事务日志记录的流发送到镜像服务器来完成，这会尽快将日志记录按顺序应用到镜像数据库中。

数据库镜像可以在不丢失已提交数据的前提下进行快速故障转移即角色切换，无须使用专门的硬件，并且容易配置和管理。

2. 运行模式

数据库镜像会话以同步操作或异步操作运行。

(1) 高性能模式(异步操作运行)：事务不需要等待镜像服务器将日志写入磁盘便可提交，这样可最大程度地提高性能。这意味着事务不需要等待镜像服务器将日志写入磁盘便可提交，而此操作允许主体服务器在事务滞后时间最小的条件下运行，但可能会丢失某些数据，此模式如图 11-21 所示。

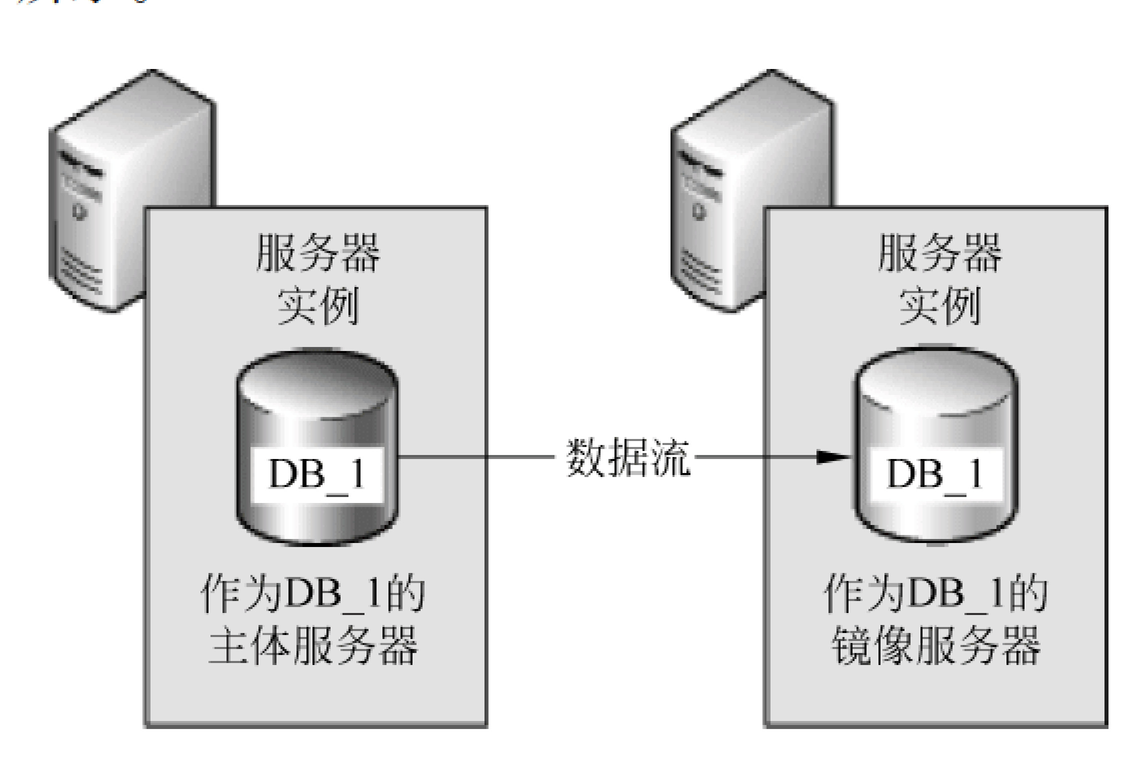


图 11-21　高性能模式的数据库镜像

(2) 高安全性模式(同步操作运行)：当会话开始时，镜像服务器使镜像数据库尽快与主体数据库同步。一旦同步了数据库，事务将在双方提交，这会延长事务滞后时间。

具有自动故障转移功能的高安全性模式需要第三个服务器实例——见证服务器，它必须与镜像服务器和主体服务器三者相互连接。与这两个伙伴服务器不同的是，见证服务器并不能用于数据库，它的作用是验证指定的伙伴服务器是否已启动并正常运行来支持自动故障转移。如果镜像服务器与主体服务器断开，但见证服务器仍与主体服务器保持连接，则镜像服务器无法启动故障转移。所以要实现故障转移的条件是主体服务器与镜像服务器断线，同时也与见证服务器断线，但镜像服务器与见证服务器保持连接的情况，此模式如图 11-22 所示。

3. 角色切换

主体服务器和镜像服务器之间的角色切换有 3 种方式，分别应用在不同条件下。

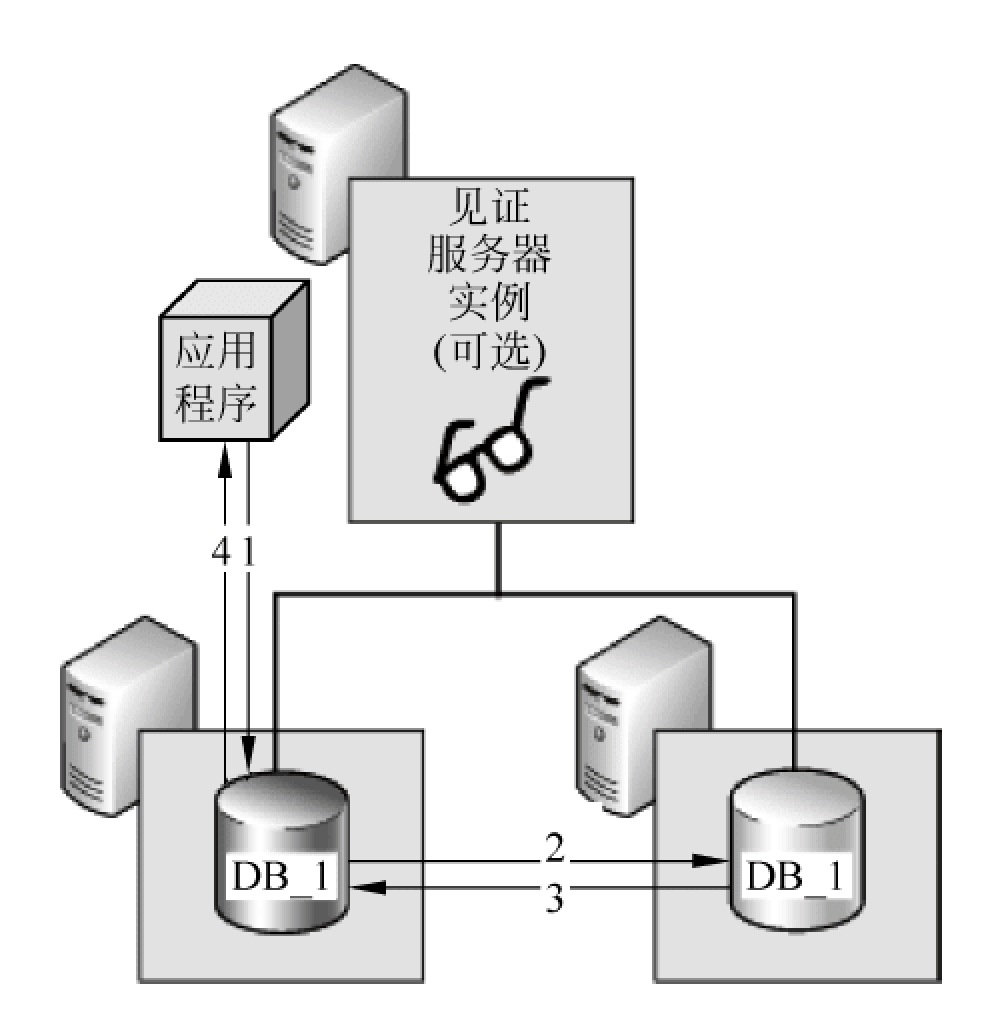


图 11-22　带见证服务器的高安全性模式的数据库镜像

(1) 自动切换：在高安全性模式及使用见证服务器的情况下，数据库必须已经同步，并且见证服务器必须和镜像服务器连接正常。如果主体服务器出现故障而镜像服务器可用，则自动故障转移，即自动切换。

(2) 手动切换：在高安全性模式下，主体服务器和镜像服务器必须保持互联，并且数据库必须已经同步。如果主体服务器出现故障而镜像服务器可用，则可手动故障转移，即手动切换。

(3) 强制服务：在高性能模式和不带自动故障转移功能的高安全性模式下，如果主体服务器出现故障而镜像服务器可用，则可以强制服务运行，相当于强制切换，这种方式可能导致某些数据丢失。

11.4.2　数据库镜像的优点

数据库镜像是一种简单的策略，具有下列优点。

1. 提高数据库的可用性

发生灾难时，在具有自动故障转移功能的高安全性模式下，自动故障转移可快速使数据库的备用副本联机(而不会丢失数据)。在其他运行模式下，数据库管理员可以选择强制服务(可能丢失数据)，以替代数据库的备用副本。

2. 增强数据保护功能

数据库镜像提供完整或接近完整的数据冗余，具体取决于运行模式是高安全性还是高性能。在 SQL Server 2008 企业版或更高版本上运行的数据库镜像伙伴会自动尝试解决某些阻止读取数据页的错误。无法读取页的伙伴会向其他伙伴请求新副本，如果此请求成功，则将以新副本替换不可读的页，这通常会消除该错误。

3. 提高生产数据库在升级期间的可用性

为了尽量减少镜像服务器的停机时间，可以按顺序升级承载故障转移伙伴的 SQL Server 实例，这样只会导致一个故障转移的停机时间。这种形式的升级称为“滚动升级”。

11.4.3 数据库镜像的配置

数据库镜像的配置相对于数据库的备份、还原以及快照等操作都要复杂得多，所以为了简化操作、方便读者理解，本书只进行高安全性模式下主体服务器和镜像服务器的配置，不配置见证服务器，所以如果主体服务器出现故障只能手动故障转移，即手动切换。

以下在 SQL Server 2008 企业版中进行数据库镜像的配置。

1. 配置环境

(1) 环境为一个 LAN 内，主体服务器 A 的 IP 地址为 222.30.251.206，服务器名为 D114-14；镜像服务器 B 的 IP 地址为 222.30.250.33，服务器名为 D114-41。

(2) 两个数据库服务器都必须设置为完整恢复模式。

(3) 两个服务器通过“证书信任”方式建立相互信任关系。

2. 配置步骤

(1) 在 A 服务器的 master 数据库中创建数据库主密钥和证书，T-SQL 代码如下：

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '1209QQ - 1'          -- 创建主密钥
GO
CREATE CERTIFICATE HOST_A_cert                                  -- 创建证书
WITH SUBJECT = 'DATABASE_MIRRORING',
START_DATE = '01/01/2017',
EXPIRY_DATE = '01/01/2019'
```

(2) 使用该证书为 A 服务器实例创建一个数据库镜像端点，T-SQL 代码如下：

```
CREATE ENDPOINT Endpoint_Mirroring                  -- 创建镜像端点，名为 Endpoint_Mirroring
STATE = STARTED                                             -- 设置端点为开启状态
AS TCP
(LISTENER_PORT = 5022,                                      -- 镜像通信的 TCP 端口号
LISTENER_IP = ALL                                           -- 允许所有 IP
)
FOR DATABASE_MIRRORING(
AUTHENTICATION = CERTIFICATE HOST_A_cert,                   -- 身份验证的证书
ENCRYPTION = REQUIRED ALGORITHM AES,                        -- 加密算法
ROLE = ALL)
```

(3) 在 SSMS 的“对象资源管理器”中，展开“服务器对象”→“端点”→“数据库镜像”选项，可以看到刚刚创建的数据库镜像端点 Endpoint_Mirroring，如图 11-23 所示。

(4) 备份 A 服务器上创建的证书，并将备份文件复制到 A 服务器上，作为两台服务器通信的凭证。备份证书的 T-SQL 语句如下：

```
BACKUP CERTIFICATE HOST_A_cert TO FILE  =  'E:\HOST_A_cert.cer'
-- 为证书的备份文件指定的名称为 HOST_A_cert.cer，存储位置为 E 盘根目录
```

(5) 在 B 服务器的 master 数据库中创建数据库主密钥和证书，并使用该证书为 B 服务器实例创建一个数据库镜像端点，最后备份 B 服务器上创建的证书，操作方法与 A 服务器



图 11-23　数据库镜像端点 Endpoint_Mirroring

相同，T-SQL 语句如下：

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '1209QQ - 1'
GO
CREATE CERTIFICATE HOST_B_cert
WITH SUBJECT = 'DATABASE_MIRRORING',
START_DATE = '01/01/2017',
EXPIRY_DATE = '01/01/2019'
GO
CREATE ENDPOINT Endpoint_Mirroring
STATE = STARTED
AS TCP
(LISTENER_PORT = 5022,
 LISTENER_IP = ALL
)
FOR DATABASE_MIRRORING(
AUTHENTICATION = CERTIFICATE HOST_B_cert,
ENCRYPTION = REQUIRED ALGORITHM AES,
ROLE = ALL)
GO
BACKUP CERTIFICATE HOST_B_cert TO FILE  =  'E:\HOST_B_cert.cer'
```

(6) 在 A 服务器创建 SQL Server 登录名和 master 数据库的数据库用户，该用户用于在镜像通信时连接 B 服务器，创建登录名和用户的 T-SQL 语句如下：

```
CREATE LOGIN HOST_B_login WITH PASSWORD  =  '123456!@#1'      -- 创建登录名
GO
CREATE USER HOST_B_user FOR LOGIN HOST_B_login                -- 创建数据库用户
```

(7) 将 B 服务器上创建的证书的备份文件复制到 A 服务器的相同位置。在 A 服务器实例上还原证书的备份文件，并将证书的使用权限授予刚刚创建的数据库用户，T-SQL 语

句如下：

```
CREATE CERTIFICATE HOST_B_cert                         -- 还原证书
AUTHORIZATION HOST_B_user                              -- 将用户账号与证书关联
FROM FILE = 'E:\HOST_B_cert.cer'                       -- B 服务器上备份过来的证书
```

(8) 将 A 服务器上的数据库镜像端点的连接权限授予刚刚创建的服务器登录名，此后两个服务器便可以使用该用户进行安全通信。授权的 T-SQL 语句如下：

```
GRANT CONNECT ON ENDPOINT::Endpoint_Mirroring TO HOST_B_login
```

(9) 在 B 服务器进行与(6)(7)(8)步相同的操作，T-SQL 语句如下：

```
CREATE LOGIN HOST_A_login WITH PASSWORD = '123456!@#2'
GO
CREATE USER HOST_A_user FOR LOGIN HOST_A_login
GO
CREATE CERTIFICATE HOST_A_cert
AUTHORIZATION HOST_A_user
FROM FILE = 'E:\HOST_A_cert.cer'
GO
GRANT CONNECT ON ENDPOINT::Endpoint_Mirroring TO HOST_A_login
```

(10) 在 A 服务器上创建 teaching 数据库的完整数据库备份，然后在 B 服务器上进行还原，从而得到初始化的镜像数据库，还原数据库时使用 NORECOVERY 模式。备份和还原数据库的 T-SQL 语句如下：

```
-- 在 A 服务器上进行的备份
BACKUP DATABASE teaching to DISK = 'E:\DATA\teaching.bak'
-- 在 B 服务器上进行的还原
RESTORE DATABASE teaching from DISK = 'E:\DATA\teaching.bak'
WITH NORECOVERY
```

(11) 在 B 服务器上配置服务器 A 为镜像伙伴，同样在 A 服务器上配置服务器 B 为镜像伙伴。至此，镜像数据库配置完成。T-SQL 语句如下：

```
-- 在 B 服务器上进行的配置
ALTER DATABASE teaching
SET PARTNER = 'TCP://222.30.251.206:5022' -- A 服务器名或 IP 地址：侦听端口
-- 在 A 服务器上进行的配置
ALTER DATABASE teaching
SET PARTNER = 'TCP://222.30.250.33:5022' -- B 服务器名或 IP 地址：侦听端口
```

在 A 服务器的对象资源管理器中右击 teaching 数据库，选择“属性”→“镜像”命令，进入 A 服务器的镜像配置窗口，可以看到镜像状态会显示为“已同步：数据库已完全同步”，如图 11-24 所示。

注意：在进行镜像同步时，需要在每一台 SQL Server 2008 服务器上启用远程管理员连接 Remote DAC，否则可能会收到错误提示。启用远程管理员连接的方法：右击服务器节点，选择“方面”命令，打开“查看方面”窗口，在“方面”下拉列表框中选择“外围应用配置器”(如图 11-25 所示)，指定“方面属性”的 RemoteDacEnabled 值为 True。

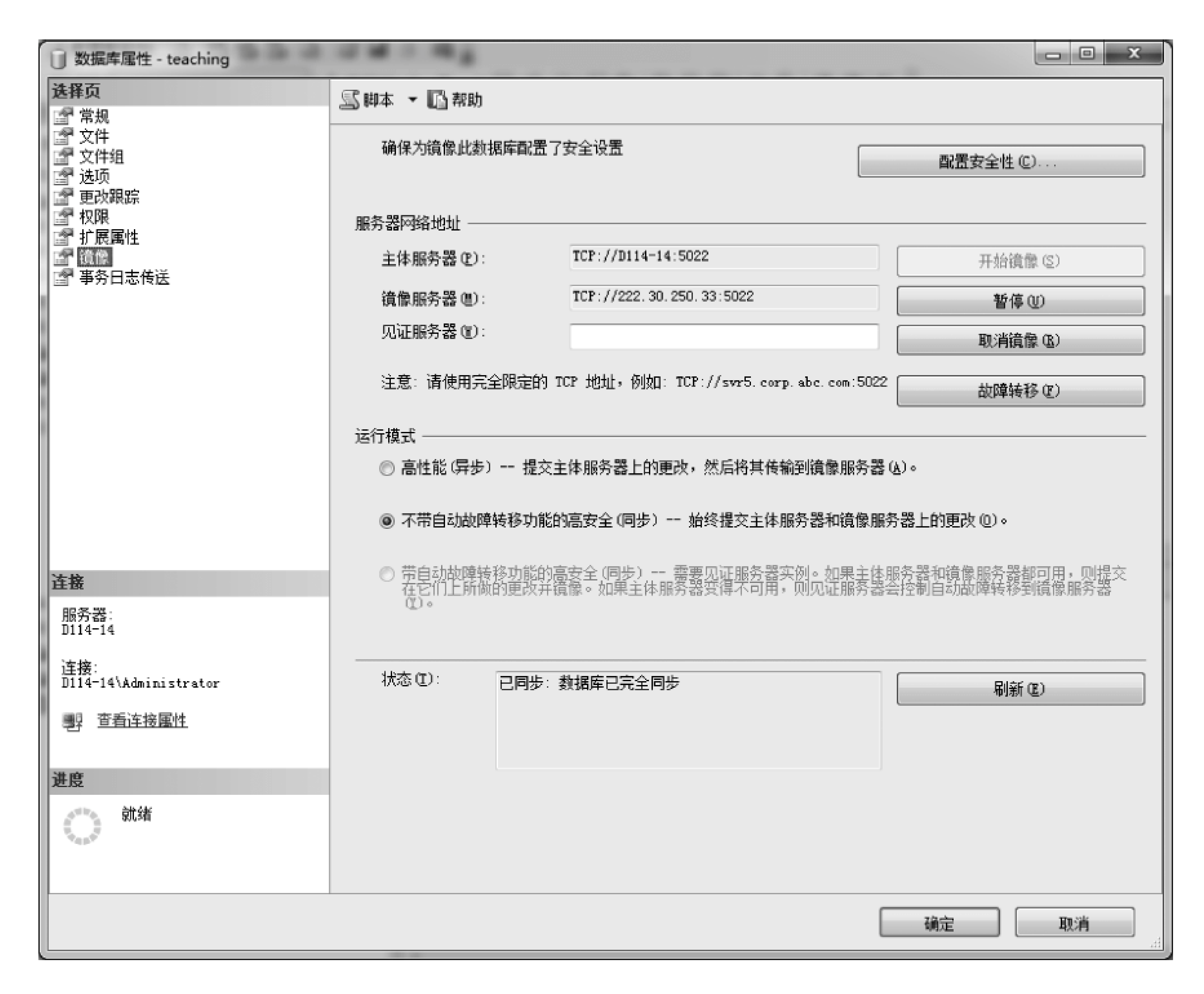


图 11-24　镜像数据库配置完成

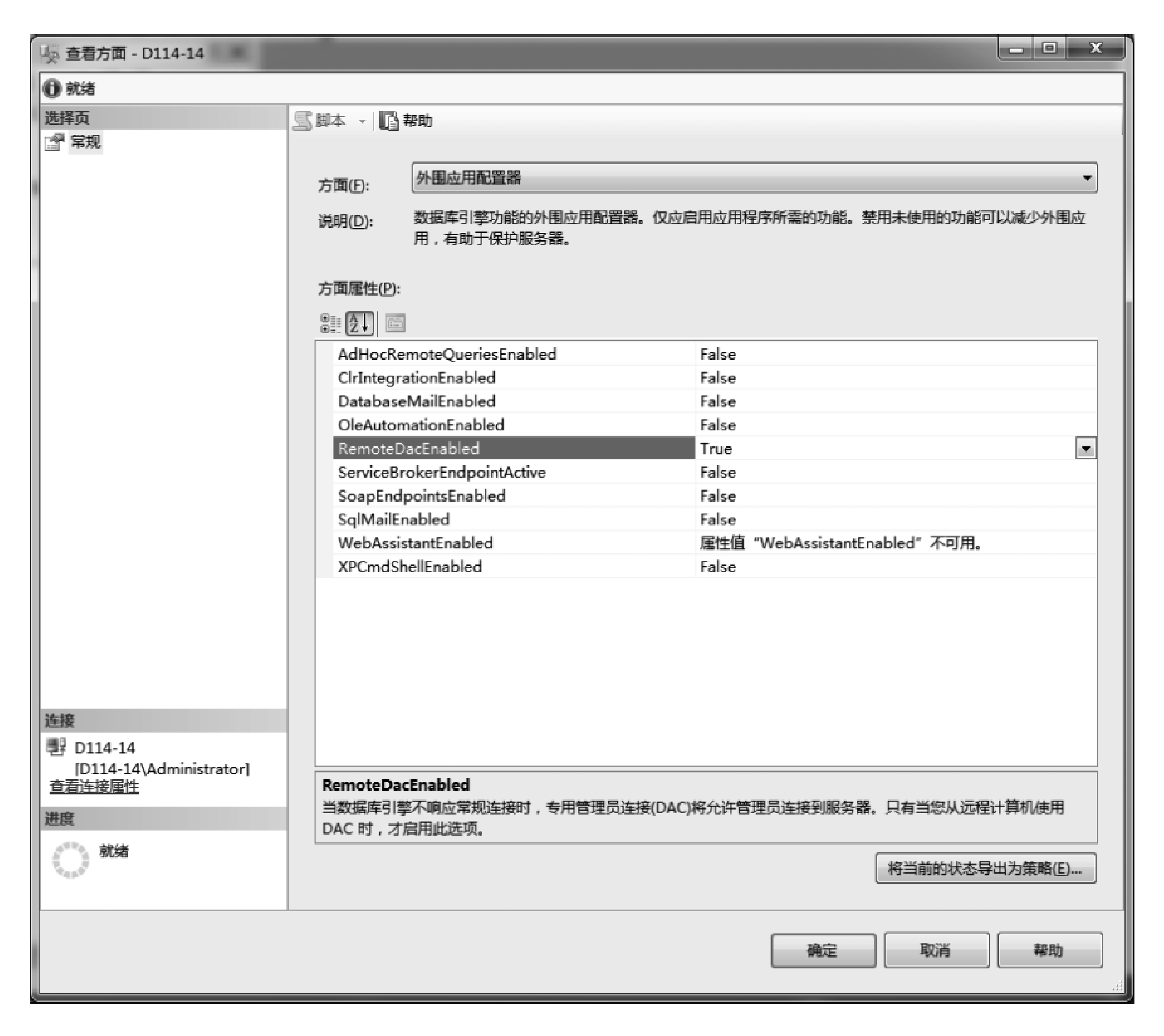


图 11-25　启用远程管理员连接

3. 实现手动故障转移

首先，确定目前主体服务器 A 和镜像服务器 B 工作均正常、连接正常，在 A 服务器上，teaching 数据库的状态是“主体，已同步”，如图 11-26 所示。



图 11-26　主体服务器状态

在 A 服务器上打开 teaching 的镜像配置窗口，确认运行模式为“高安全”模式，然后单击“故障转移”按钮，弹出“数据库属性”对话框，提示是否进行转移，如图 11-27 所示，单击“是”按钮进行转移。

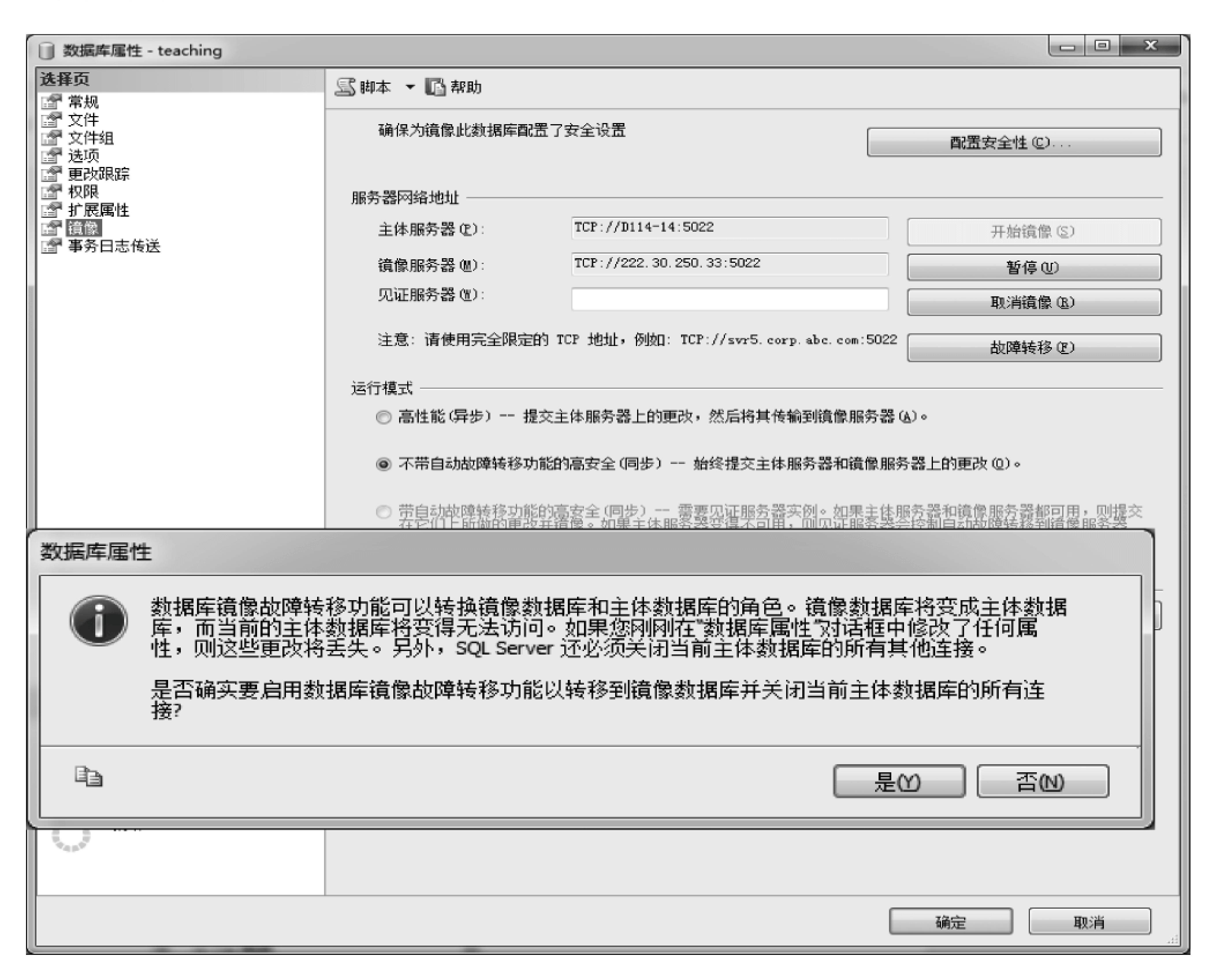


图 11-27　手动故障转移

成功完成操作后，在 A 服务器上数据库 teaching 的状态变为了“正在还原...”，如图 11-28 所示。

在 B 服务器上数据库 teaching 的状态变为“主体，已同步”，如图 11-29 所示。可见，故障转移的结果使主体和镜像角色互换了。





图 11-28　A 服务器的镜像状态



图 11-29　B 服务器的主体状态

关于自动切换和强制服务方式的配置这里不再讲述，请读者自行学习。

习　题　11

1. 简述数据库备份和还原的基本概念。
2. 数据库备份有哪几种类型？
3. 简述数据库的恢复模式。
4. 简述在 SSMS 中使用界面进行备份和还原数据库 bankcard 的操作过程。
5. 使用 T-SQL 语句分别实现数据库 bankcard 的备份和还原操作。
6. 了解数据库的分离和附加的作用及操作方法。
7. 什么是数据库快照？
8. 简述数据库快照的优点。
9. 创建 bankcard 数据库快照，并恢复。
10. 什么是数据库镜像？
11. 简述主体服务器和镜像服务器之间角色切换的 3 种方式。
12. 简述数据库镜像的优点。
13. 配置 bankcard 数据库的镜像，并进行手动切换。

第 12 章　数据库复制

SQL Server 2008 中的复制(Replication)是 SQL Server 高可用性的核心功能之一,它指的并不仅仅是一项技术,而是一系列技术的集合,包括从存储转发数据到同步数据到维护数据一致性。复制主要用于解决分布式数据问题。本章首先简要介绍 bcp 导入导出数据,然后重点介绍复制的基本概念、工作机制和配置操作。

12.1　bcp 导入导出数据

大容量复制程序(bulk copy program,简称 bcp)是 SQL Server 提供的一个快捷的数据导入导出工具,比使用 SSMS 提供的数据库导入导出要高效得多。如果用 SSMS 提供的导入导出对大容量数据进行迁移,其速度是人们无法接受的。

12.1.1　bcp 命令简介

bcp 是 SQL Server 中负责导入导出数据的一个命令行工具,它是基于 DB-Library 的,并且能以并行的方式高效地导入导出大批量的数据。bcp 可以将数据库的表或视图直接导出,也可以通过 SELEC 语句对表或视图进行过滤后导出。在导入导出数据时,可以使用默认值或使用一个格式文件将文件中的数据导入到数据库或将数据库中的数据导出到文件中。

bcp 命令的语法格式如下:

```
bcp {[[database_name.][owner].]{table_name | view_name} | "query"}
    {in | out | queryout | format} data_file
    [-m max_errors] [-f format_file] [-x] [-e err_file]
    [-F first_row] [-L last_row] [-b batch_size]
    [-n] [-c] [-w] [-N] [-V (70|80|90|110)]
    [-q] [-C code_page] [-t field_term] [-r row_term]
    [-i input_file] [-o output_file] [-a packet_size]
    [-S server_name[\instance_name]] [-U login_id] [-P password]
    [-T] [-v] [-R] [-k] [-E] [-h "hint [,...n]"]
```

其中的主要参数介绍如下:

(1) database_name——制定表或视图所在的数据库名,省略则为用户的默认数据库。

(2) owner——表或视图所有者的名称。如果操作者即是拥有者,则可省略。

(3) table_name | view_name——导出数据时的源表或视图名;导入数据时的目标表或视图名。

(4) query——SELECT 查询语句,返回查询结果集作为导出时的数据源。

(5) in——bcp 命令为实现大容量导入,后面跟需要导入的文件名。

(6) out——bcp 命令为实现大容量导出,后面跟需要导出到的文件名。

(7) queryout——bcp 命令为实现使用 SQL 语句导出,它与 out 类似,只是数据源不是表或视图名,而是 SQL 语句。

(8) format——导出格式文件,后面跟格式文件名。

(9) data_file——导入时的源文件名或导出到的目标文件名,包含文件的完整路径。

(10) -m max_errors——指定取消 bcp 操作之前可能出现的语法错误的数目,语法错误是指将数据转换为目标数据类型时的错误,默认值为 10。

(11) -f format_file——format_file 表示格式文件名。这个选项依赖于上述的动作,如果使用的是 in 或 out,则表示已经存在的格式文件;如果使用的是 format,则表示是要生成的格式文件。

(12) -x——这个选项要和-f format_file 配合使用,以便生成 xml 格式的格式文件。

(13) -e err_file——指定错误文件名及其完整路径,此文件用于存储 bcp 无法从文件传输到数据库的所有行。若省略,则不创建错误文件。

(14) -F first_row——指定从被导出表的哪一行导出,或从被导入文件的哪一行导入。省略此项,则从第一行导入或导出。

(15) -L last_row——指定被导出表要导到哪一行结束,或从被导入文件导数据时,导到哪一行结束。省略此项,则为最后一行。

(16) -b batch_size——指定每批导入的数据行数。默认情况下,所有数据行均作为一批导入。

(17) -n——使用数据的本机数据类型执行大容量复制操作。

(18) -c——使用 char 类型作为存储类型,没有前缀且以"\t"作为字段分隔符,以"\n"作为行分隔符。

(19) -w——和-c 类似,只是当使用 Unicode 字符集复制数据时使用,且以 nchar 作为存储类型。

(20) -N——执行大容量复制操作时,对字符数据使用 Unicode 字符,它是-w 选项的一个替代项,性能更高。

(21) -V (70|80|90|110)——使用 SQL Server 早期版本中的数据类型执行大容量复制操作。

(22) -q——使用此选项可以指定包含空格或单引号的数据库、所有者、表或视图的名称。

(23) -C code_page——指定数据文件中数据的代码页。

(24) -t field_term——指定字段分隔符,默认是"\t"。

(25) -r row_term——指定行分隔符,默认是"\n"。

(26) -S server_name[\instance_name]——指定要连接的 SQL Server 服务器实例,如果未指定此选项,则 bcp 连接本机的 SQL Server 默认实例。如果要连接某台机器上的默认实例,只需要指定机器名即可。

(27) -U login_id——指定连接 SQL Sever 的用户名。

(28) -P password——指定连接 SQL Server 的用户名密码。

(29) -T——指定 bcp 使用信任连接登录 SQL Server。如果未指定-T，则必须指定-U和-P。

(30) -v——报告 bcp 使用工具的版本号和版权。

(31) -k——指定空列使用 null 值插入，而不是这列的默认值。

(32) -E——指定导入数据文件中的标识值用于标识列。

(33) -h "hint [,...n]"——指定向表或视图中大容量导入数据时所用的提示。

12.1.2 bcp 导出

bcp 可以从整表、视图或查询中导出数据，导出时必须指定目的文件名，如果文件已存在，则被覆盖。导出数据可以使用密码连接，也可以使用信任连接；可以在 Windows 控制台执行，也可以以 SQL 语句的方式运行。

1. 导出整个表或视图

【例 12-1】 将 teaching 数据库中 student 表的数据导出到 stu. txt 文件中。

在 Windows 控制台执行的两种数据库连接方式如下：

```
bcp teaching.dbo.student out e:\stu.txt -c -S YINZHIYU-PC\SQLEXPRESS -U"u2" -P"u2"
--使用密码连接
```

注意：将-U 后的用户名和-P 后的密码加上双引号，用户 u2 必须拥有 student 表的 select 权限。

```
bcp teaching.dbo.student out e:\stu.txt -c -S YINZHIYU-PC\SQLEXPRESS -T
--使用信任连接
```

我们在 SQL Server 安装目录下执行"密码连接方式"的代码，执行结果如图 12-1 所示。

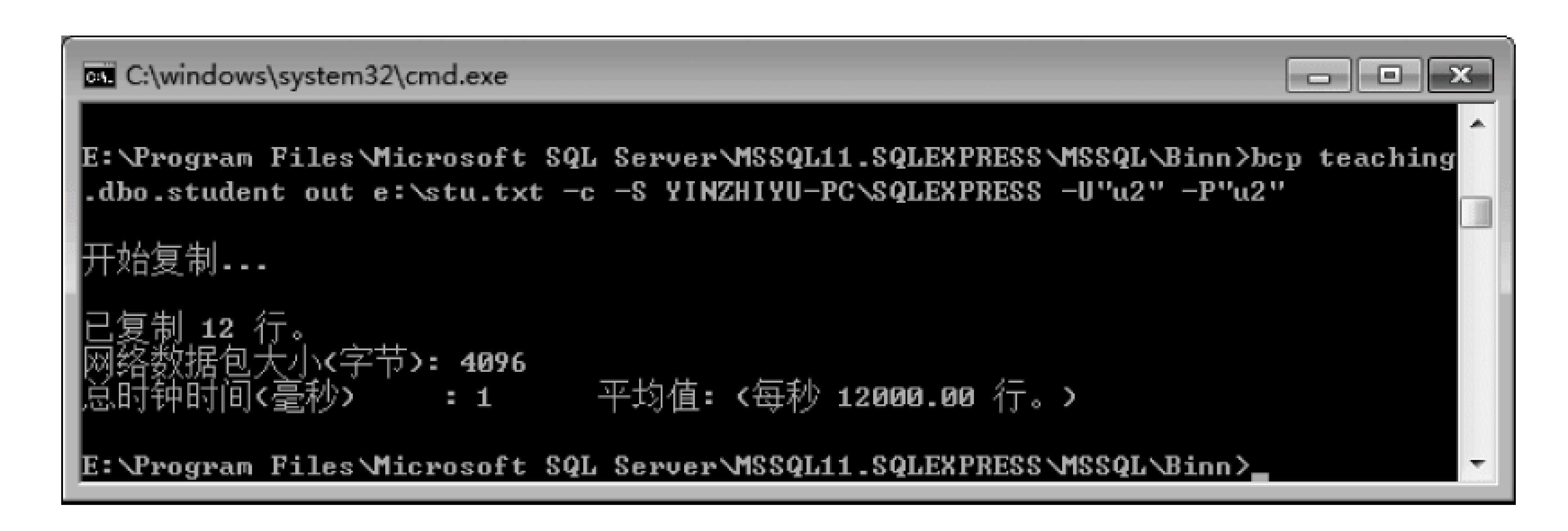


图 12-1 将 student 表的数据导出到 stu. txt 文件

导出到 stu. txt 文件的内容如图 12-2 所示。

bcp 除了可以在控制台执行外，还可以通过调用 SQL Server 的一个系统存储过程 xp_cmdshell 以 SQL 语句的方式运行 bcp。如上述第一条命令可改写为：

```
EXEC master..xp_cmdshell 'bcp teaching.dbo.student out e:\stu.txt -c -S YINZHIYU-PC\
SQLEXPRESS -U"u2" -P"u2"'
```

在 master 数据库下执行时，master..可省略。运行结果如图 12-3 所示。

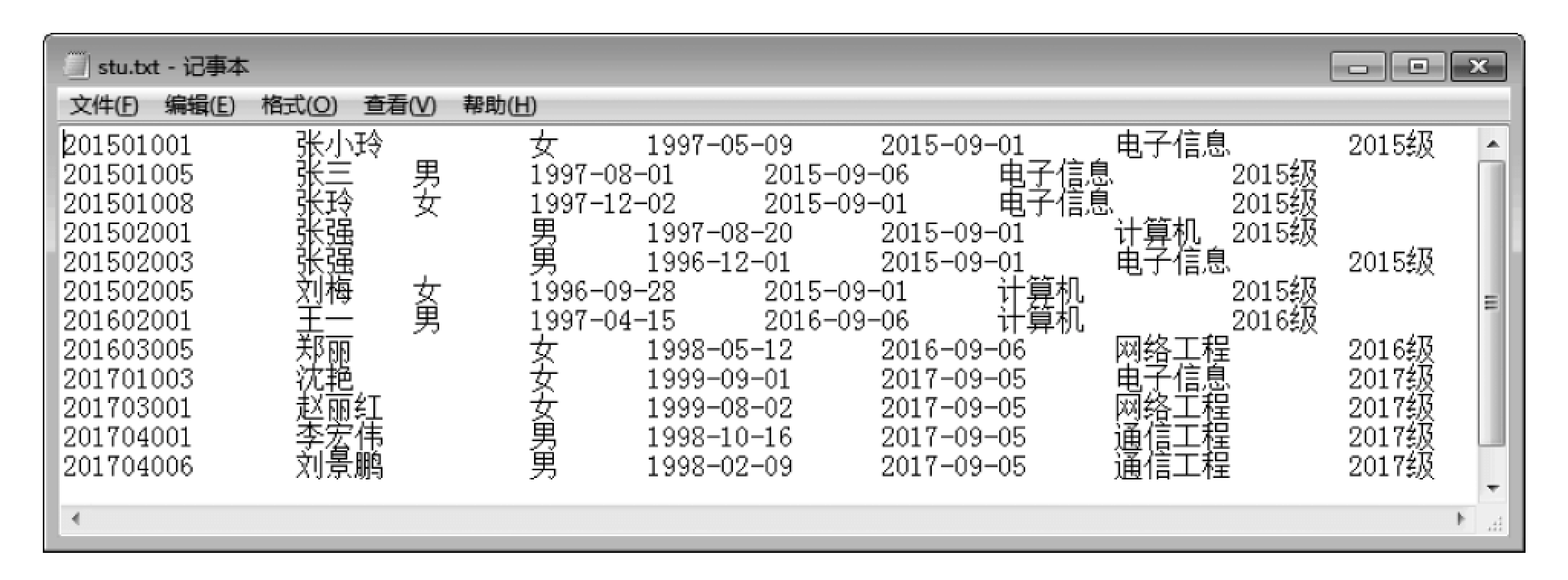


图 12-2　stu.txt 文件的内容

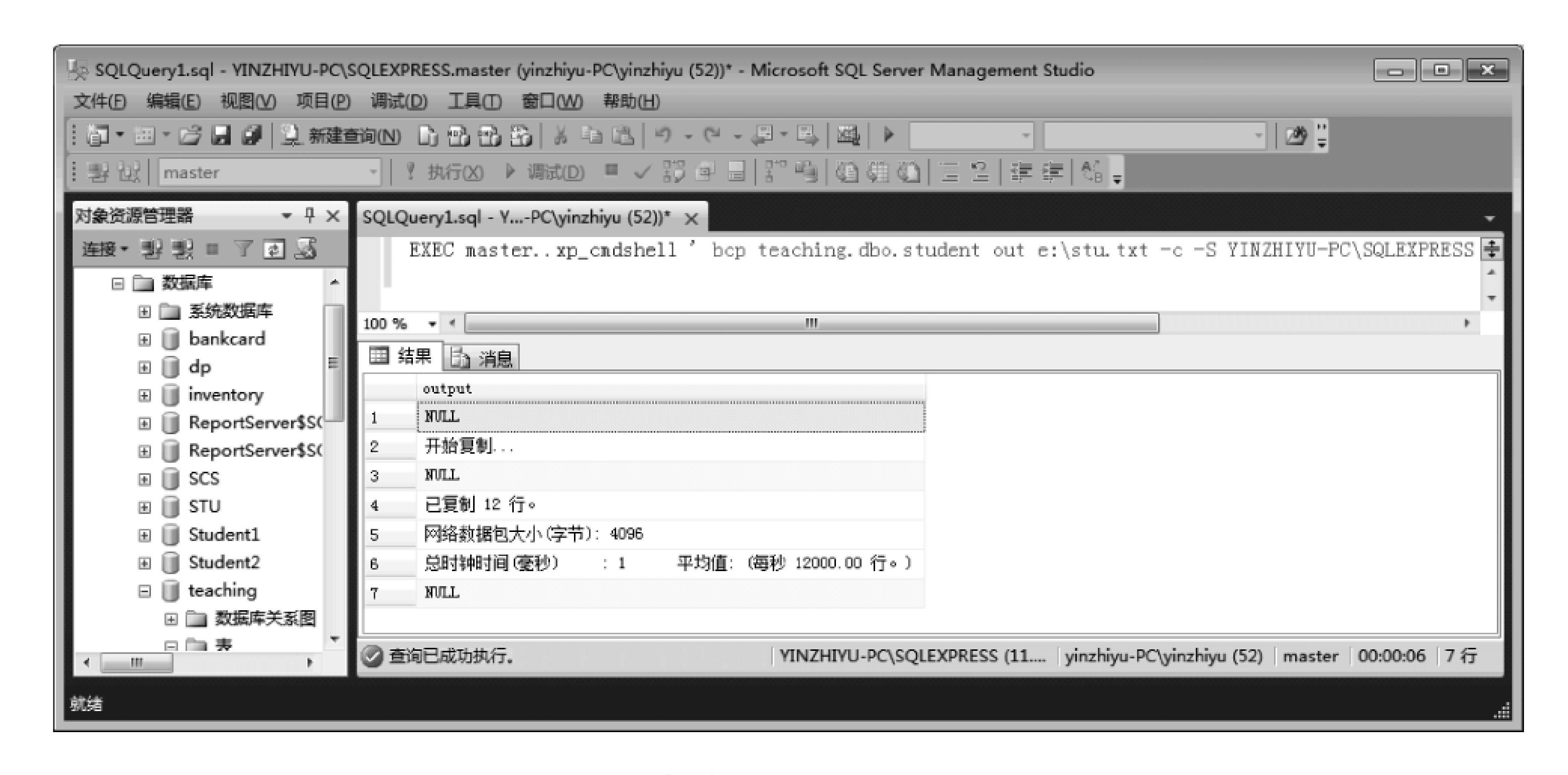


图 12-3　通过系统存储过程 xp_cmdshell 执行 bcp 导出

2. 导出表的查询结果

bcp 不仅可以接受表名或视图名作为参数，也可以接受 SQL 查询作为参数。通过 SQL 语句可以对要导出的表进行过滤，然后导出过滤后的记录。

【例 12-2】 将 teaching 数据库中 student 表的男生数据导出到 stu1.txt 文件中。

```
bcp "select * from teaching.dbo.student where ssex = '男'" queryout e:\stu1.txt -c -S
YINZHIYU-PC\SQLEXPRESS -U"u2" -P"u2"
```

执行结果如图 12-4 所示。



图 12-4　将 student 表的男生数据导出到 stu1.txt 文件

bcp 还可以通过简单地设置选项对导出的行进行限制。

下面的 bcp 命令使用了两个参数：-F 3 和-L 5，表示从 select * from teaching.dbo.student where ssex='男'所查出来的结果中取第 3 条到第 5 条记录进行导出。

```
bcp "select * from teaching.dbo.student where ssex = '男' " queryout e:\stu2.txt -F 3 -L 5 -c
-S YINZHIYU-PC\SQLEXPRESS -U"u2" -P"u2"
```

控制台执行的信任数据库连接方式以及 SQL 语句的方式运行 bcp 导出表的查询结果，读者可以自行操作实现，这里不再讲述。

3. bcp 导出格式文件

bcp 不仅可以根据表、视图导入导出数据，还可以配合格式文件对导入导出数据进行限制。格式文件以纯文本文件形式存在，分为一般格式和 xml 格式。用户可以手工编写格式文件，也可以通过 bcp 命令根据表、视图自动生成格式文件。

【例 12-3】 将 student 表的结构生成了一个普通格式文件 stu_format1.fmt，不导出 student 表的内容，所以导出到的目标文件名为 nul。本例采用控制台执行，使用信任连接方式。

```
bcp teaching.dbo.student format nul -f e:\stu_format1.fmt -c -S YINZHIYU-PC\ SQLEXPRESS
-T
```

通过记事本打开 stu_format1.fmt 文件，这个格式文件的内容如图 12-5 所示。

```
stu_format1.fmt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
9.0
7
1       SQLCHAR     0     9      "\t"      1     sno            Chinese_PRC_CI_AS
2       SQLCHAR     0     16     "\t"      2     sname          Chinese_PRC_CI_AS
3       SQLCHAR     0     2      "\t"      3     ssex           Chinese_PRC_CI_AS
4       SQLCHAR     0     20     "\t"      4     birthday       Chinese_PRC_CI_AS
5       SQLCHAR     0     20     "\t"      5     en_time        Chinese_PRC_CI_AS
6       SQLCHAR     0     20     "\t"      6     specialty      Chinese_PRC_CI_AS
7       SQLCHAR     0     10     "\r\n"    7     grade          Chinese_PRC_CI_AS
```

图 12-5　stu_format1.fmt 文件的内容

这个格式文件记录了这个表的字段类型、长度、字段分隔符、行分隔符和字段名等信息。

bcp 还可以通过-x 选项生成 xml 格式的格式文件。

【例 12-4】 将 student 表的结构生成了一个 XML 格式文件 stu_format2.fmt。

```
bcp teaching.dbo.student format nul -f e:\stu_format2.fmt -x -c -S YINZHIYU-PC\
SQLEXPRESS -T
```

xml 格式文件所描述的内容和普通格式文件所描述的内容完全一样，只是格式不同，如图 12-6 所示。

12.1.3　bcp 导入

bcp 导入数据是导出数据的反操作，可以通过 in 命令将上面所导出的 stu.txt 或 stu1.txt 再重新导入到数据库中。

```
<?xml version="1.0"?>
<BCPFORMAT xmlns="http://schemas.microsoft.com/sqlserver/2004/bulkload/format" xmlns:xsi="http://www.w
 <RECORD>
  <FIELD ID="1" xsi:type="CharTerm" TERMINATOR="\t" MAX_LENGTH="9" COLLATION="Chinese_PRC_CI_AS"/>
  <FIELD ID="2" xsi:type="CharTerm" TERMINATOR="\t" MAX_LENGTH="16" COLLATION="Chinese_PRC_CI_AS"/>
  <FIELD ID="3" xsi:type="CharTerm" TERMINATOR="\t" MAX_LENGTH="2" COLLATION="Chinese_PRC_CI_AS"/>
  <FIELD ID="4" xsi:type="CharTerm" TERMINATOR="\t" MAX_LENGTH="20" COLLATION="Chinese_PRC_CI_AS"/>
  <FIELD ID="5" xsi:type="CharTerm" TERMINATOR="\t" MAX_LENGTH="20" COLLATION="Chinese_PRC_CI_AS"/>
  <FIELD ID="6" xsi:type="CharTerm" TERMINATOR="\t" MAX_LENGTH="20" COLLATION="Chinese_PRC_CI_AS"/>
  <FIELD ID="7" xsi:type="CharTerm" TERMINATOR="\r\n" MAX_LENGTH="10" COLLATION="Chinese_PRC_CI_AS"/>
 </RECORD>
 <ROW>
```

图 12-6　stu_format2.fmt 文件的内容

【例 12-5】 首先创建一个和 student 表结构完全相同的 student1 表，然后将 stu1.txt 文件中的数据导入到 student1 表中。本例采用控制台执行，使用信任连接方式。

```
bcp teaching.dbo.student1 in e:\stu1.txt -c -S YINZHIYU-PC\SQLEXPRESS -T
```

执行结果如图 12-7 所示。



图 12-7　stu1.txt 文件中数据导入到 student1 表

导入数据也同样可以使用-F 和-L 选项来选择导入数据的记录行，比如将 stu1.txt 文件中第 3 条到第 5 条记录导入到 student1 表中，代码如下：

```
bcp teaching.dbo.student1 in e:\stu1.txt -F 3 -L 5 -c -S YINZHIYU-PC\SQLEXPRESS -T
```

在向一个数据库表导入数据时，可以先导出其格式文件，然后根据已经存在的格式文件将满足条件的记录导入，不满足则不导入。比如，上述的格式文件 stu_format1.fmt 中的第二个字段的字符长度是 16，如果某个文本文件中的相应字段的长度超过 16，则这条记录将不被导入到数据库中，其他满足条件的记录正常导入。

在例 12-5 代码中加入普通格式文件 stu_format1.fmt 限制，代码如下：

```
bcp teaching.dbo.student1 in e:\stu1.txt -c -S YINZHIYU-PC\SQLEXPRESS -f e:\stu_format1.
fmt -T
```

12.2　复　　制

bcp 是 SQL Server 提供的一个快捷的大容量数据导入导出工具，但操作不够自动。在自动的数据同步和批量操作上，SQL Server 中的复制提供了强大的功能支持。

12.2.1 复制简介

对于一个地域分散的大型企业组织来说，构建具有典型的分布式计算机特征的大型企业管理信息系统时，总要解决一个很重要的问题：如何在多个不同数据库服务器之间保证共享数据的一致性。因为企业组织会在不同地点对具有相同结构的本地数据库进行修改，但要保证修改后的数据库有相同的结果，其本质就是在对本地数据库的修改要体现在其他具有相同数据的远程数据库中。那么如何解决这个问题就要用到数据库的复制技术。

SQL Server 2008 提供了内置的复制能力，复制组件并不是附加产品，而是核心引擎的一部分。在复制这个技术的支持下，用户可以维护网络上不同数据库服务器上数据的多个副本，从而保证数据的一致性。

我们在第 11 章介绍过数据库同步备份的另一种方式：SQL Server 的数据库镜像。数据库镜像的优点是系统能自动发现主服务器故障，并且可以自动切换至镜像服务器。但缺点是配置复杂，镜像数据库中的数据不可见(在 SSMS 中，只能看到镜像数据库处于镜像状态，无法进行任何数据库操作，最简单的查询也不行，想看看镜像数据库中的数据是否正确都不行。只有将镜像数据库切换为主数据库才可见)。

本章要介绍的数据库复制，所有同步的数据库中数据都是可见而且可用的。

SQL Server 2008 使用发布和订阅这一术语来描述其复制活动。所谓发布，就是向其他数据库服务器(订阅者)复制数据，订阅就是从另外的服务器(发布者)接收复制数据。虽然发布和订阅的对象都是将数据复制，但发布和订阅并不是不同角度的同一数据操作，而是体现出一定的层次性和顺序性，总是先进行发布，再进行订阅。复制的拓扑结构如图 12-8 所示。

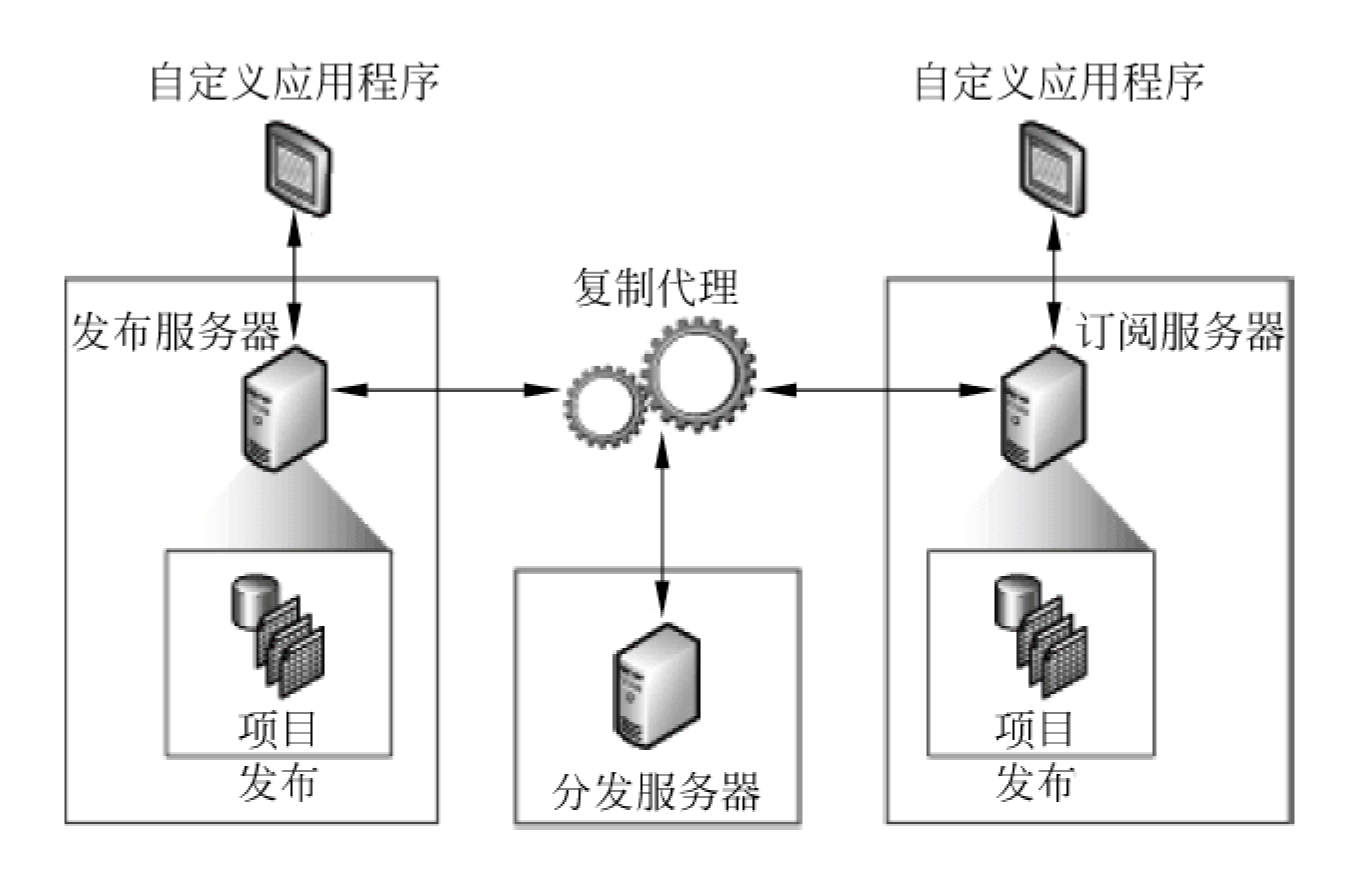


图 12-8 复制的拓扑结构

SQL Server 的复制组件有项目、论文、发布者、分发者、订阅者、推订阅和拉订阅等等。

1. 项目和论文

论文是被复制的数据集合，一篇论文一般就是咱们所说的表，论文是项目的基本组成单元。项目是论文的集合，它可以包括一个或多个论文，订阅者订阅的是项目而不是项目中的论文，这样可以使订阅更为简单。

2. 发布者

发布者是发布项目的服务器。发布者服务器维护源数据库以及有关项目的信息，使数据可用于复制。除了决定哪些数据将被复制外，发布者还要检测哪些复制数据发生了变化，并将这些变化复制到分发者的分发数据库中。

3. 分发者

分发者是发布者和订阅者之间的桥梁，起着存储区的作用。它负责维护分发数据库，把从发布者传递来的复制数据传送至相应的订阅者的服务器，并且在某些情况下，为从发布者向订阅者移动的数据排队。

4. 订阅者

订阅者是指存储复制数据的副本机器，接收并维护已发布的数据，订阅者也可以对发布的数据进行修改。尽管订阅者可以对数据进行修改，但它仍是一个订阅者，当然订阅者也可以作为其他订阅者的发布者。

发布者、分发者、订阅者实际上并不一定指相互独立的服务器。它只是对 SQL Server 在复制过程中所扮演的不同角色的描述。SQL Server 允许一台 SQL Server 服务器扮演不同的角色，比如一台发布者的服务器既可发布项目，也可以作为分发者来存储和传送复制。在实际应用中，用户决定是否让一台服务器扮演一个或多个角色。大型数据库系统基于对复制系统性能的考虑，为了提高分发者从分发数据库向订阅者的数据库复制项目的效率，降低发布者服务器的负载，用户常常不允许一台服务器既扮演发布者又扮演分发者，而是让其他的服务器专门承担分发者任务，从而提高了发布者和分发者的性能。

5. 推订阅和拉订阅

推订阅是指由发布者将所有发生在发布数据库的修改复制给订阅者，而不必向订阅者发出请求，只要发布数据库发生修改，发布者就会自动把这种修改体现在订阅者那里。在对数据同步性要求比较高的场合，最好使用推订阅。拉订阅是指由订阅者在经过一段时间就会向发布者要求复制发布数据库发生的变化。

12.2.2 复制的类型

SQL Server 2008 提供了 3 种复制类型：快照复制、事务复制和合并复制。可以在实际应用中使用相应的复制类型，每一种复制类型都在不同程度上实现数据的一致性。

1. 快照复制

顾名思义，快照复制指在某一时刻给发布数据库中的发布数据创建快照，然后将数据复制到订阅者服务器。快照复制实现较为简单，其所复制的只是某一时刻数据库的瞬间数据。

快照复制是将整个项目传送给订阅者，就是在某一时刻将发布数据创建一次快照，生成一个描述发布数据库中数据当前状态的一个文件，然后在相应的时间将其复制到订阅者的数据库上。快照复制并不是不停地监视发布数据库中发生的变化情况，它是对发布数据库进行一次扫描，把所有发布数据中的数据从源数据库送至目标数据库，而不仅仅是变化的数据。如果数据量很大，那么要复制的数据就很多。因此对网络资源要求很高，不仅要有较快的传输速度，而且要保证传输的可靠性。

快照复制是最为简单的一种复制类型，能够在发布者和订阅者之间保证数据的一致性。

快照复制通常使用在以下场合：在一定时间内出现大量的更改的操作，但数据总量不大，变化周期较长。

2. 事务复制

快照复制是将整个数据集发送给订阅服务器，由于体积大而造成复制周期较长，会形成复制滞后问题。那么事务复制使用事务日志来生成将复制到订阅服务器的事务，因为它只复制事务也就是变化，所以滞后也比快照复制低得多，在订阅服务器处将得到及时应用。

事务复制有3个组件：

- 快照代理——它生成架构、数据以及跟踪复制过程所需的数据；
- 分发代理——它分发快照和随后的命令；
- 日志读取器代理——它读取发布数据的事务日志。

在事务复制中，当发布数据库发生变化时，这种变化就会立即传递给订阅者，并在较短时间内完成(一般为几秒)，而不是像快照复制那样要经过很长一段时间间隔。因此，事务复制是一种接近实时地从源到目标分发数据的方法。由于某种原因，事务复制的频率较高，所以必须保证在订阅者与发布者之间要有可靠的网络连接。

3. 合并复制

合并复制是为移动用户设计的，可以在发布服务器或是订阅服务器处执行修改，在合并代理运行时，这些修改将同步，多用于发布服务器与订阅服务都修改数据的情况下。其工作原理如下：在要复制的每个表上实现触发器，并使用包含GUID的列唯一标识要复制的表中的每一行。对其中的任何一个表进行修改时，都会将更改记录到一个数据表中，在合并代理运行时，它收集数据表中的GUID，这些GUID指出了在发布服务器和订阅服务器处修改过的行。对于只在发布服务器或是订阅服务器修改的数据则直接进行相应操作，如INSERT、UPDATE、DELETE，如果双方都有GUID，则按照用户指定的方式解决冲突，默认发布服务器优先。

12.2.3 配置复制

无论是快照复制、事务性复制还是合并复制，创建复制都要经过相似的几个步骤，一般用向导来完成，以下以快照复制为例进行介绍。

1. 复制准备

这里的Windows版本是Windows 7系统，选择发布服务器和分发服务器，发布服务器名为D114-11，订阅服务器名为D114-12。

不论是发布服务器还是订阅服务器都要开启代理服务。

(1) 打开SSMS，在“对象资源管理器”窗口中的目标服务器上右击，弹出快捷菜单，选择“属性”命令。

(2) 出现“服务器属性”窗口，选择“选择页”中的“安全性”选项，进入安全性设置页面，选择“启用服务器代理账户”单选按钮，设置账户名为sa，并输入密码，单击“确定”按钮。

(3) 在SSMS的“对象资源管理器”中，右击最下面的“SQL Server代理”，选择“开启”命令。

2. 创建发布服务器和分发服务器

如果有条件，可以单独创建一个分发服务器，这里就将发布服务器和分发服务器设置在

同一台计算机上。

(1) 在 SSMS 的“对象资源管理器”中，展开“复制”节点，右击“本地发布”，选择“新建发布”命令，出现“新建发布向导”，如图 12-9 所示。



图 12-9　新建发布向导

(2) 单击“下一步”按钮，设置分发服务器，如图 12-10 所示。



图 12-10　设置分发服务器

(3) 单击"下一步"按钮,将 SQL Server 代理配置为自动启动,如图 12-11 所示。

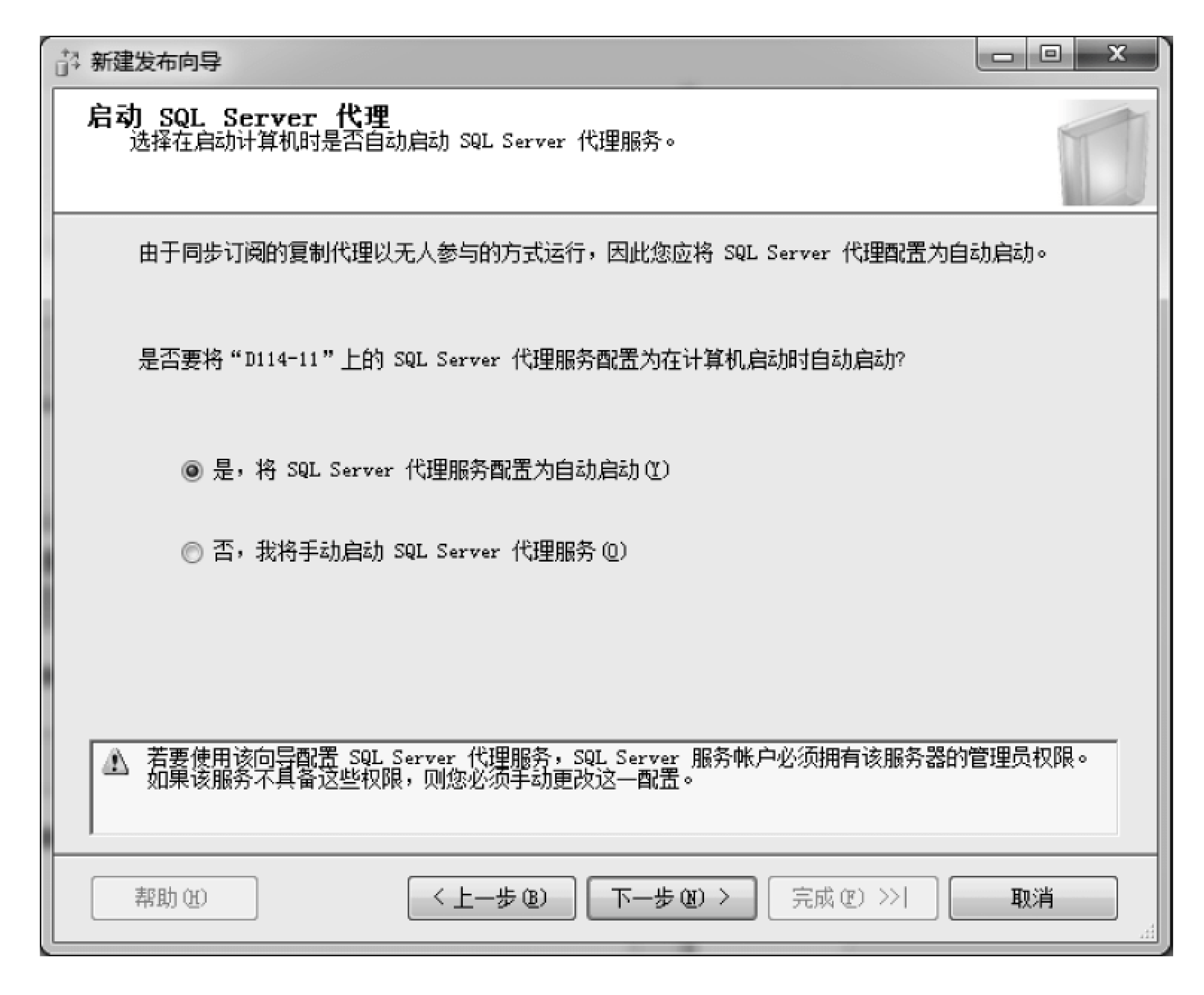


图 12-11　启动 SQL Server 代理

(4) 单击"下一步"按钮,指定存储快照的文件夹,如图 12-12 所示。



图 12-12　指定快照文件夹

(5) 单击"下一步"按钮,选择要发布的数据库,比如 teaching 数据库,如图 12-13 所示。

(6) 单击"下一步"按钮,选择发布类型,这里选择"快照发布",如图 12-14 所示。

图 12-13　选择要发布的数据库



图 12-14　选择发布类型

(7) 单击“下一步”按钮，选择要作为项目发布的数据库对象，这里选择了其中的两个表，如图 12-15 所示。

(8) 单击“下一步”按钮，出现“筛选表行”窗口，如图 12-16 所示。可以选择添加筛选



图 12-15　选择作为项目发布的数据库对象



图 12-16　筛选表行

器，通过设置筛选语句(select)筛选表中的行，如图 12-17 所示；也可以不添加，直接进行下一步。

(9) 单击“下一步”按钮，打开“快照代理”窗口，指定何时创建快照，对订阅进行初始化，这里选择的是立即创建，如图 12-18 所示。



图 12-17 添加筛选器



图 12-18 创建快照对订阅初始化

(10) 单击“下一步”按钮，打开“代理安全性”窗口，进行安全设置，如图 12-19 所示。

(11) 单击“安全设置”按钮。首先，为快照代理选择登录名，这里选择“在 SQL Server 代理服务账户下运行”单选按钮；然后，为连接到发布服务器设置登录名 sa，如图 12-20 所示。



图 12-19　代理安全性设置



图 12-20　选择执行快照代理的 Windows 账户

(12) 单击“下一步”按钮，可以选择创建发布和生成脚本文件选项，这里只选择“创建发布”复选框，如图 12-21 所示，单击“完成”按钮。

(13) 单击“下一步”按钮，为新建的发布取名，如图 12-22 所示。

(14) 单击“完成”按钮，创建发布 teaching 成功，如图 12-23 所示。



图 12-21　选择向导结束时创建发布



图 12-22　为新建的发布取名为 teaching

3. 创建订阅服务器

(1) 在 SSMS 的“对象资源管理器”中,展开“复制”节点,右击“本地订阅”,选择“新建订阅”命令,出现“新建订阅向导”,如图 12-24 所示。



图 12-23 创建发布 teaching 完成



图 12-24 新建订阅向导

(2) 单击“下一步”按钮，选择发布服务器为 D114-11 在“数据库和发布”列表框中选择数据库 teaching 下的发布 teaching，如图 12-25 所示。

(3) 单击“下一步”按钮，选择分发代理位置，这里选择“在分发服务器 D114-11 上运行所有代理(推送订阅)”，如图 12-26 所示。



图 12-25　选择发布

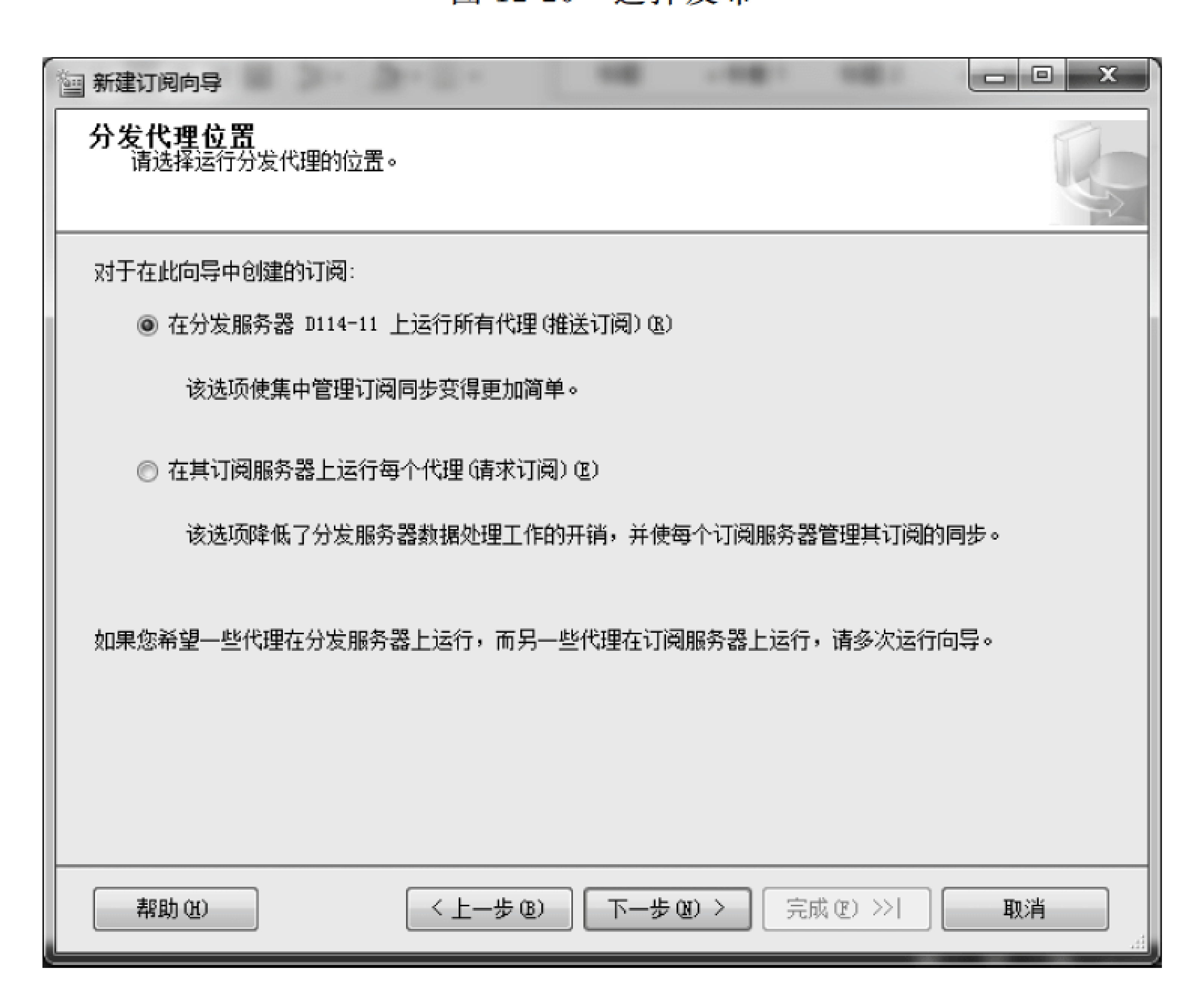


图 12-26　选择分发代理位置

(4) 单击“下一步”按钮，选择订阅服务器和订阅数据库，这里选择订阅服务器为 D114-12，新建一个用于订阅的数据库，命名为 teaching1，如图 12-27 所示。

(5) 单击“下一步”按钮，进行分发代理安全性设置。单击“订阅属性”右侧的按钮，进入如图 12-28 所示的对话框，首先将“指定在同步此订阅时运行分发代理进程的域或计算机账户”设置为“在 SQL Server 代理账户下运行”；然后将“连接到分发服务器”设置为“通过模拟进程账户”；最后，在“连接到订阅服务器”选项组中将 SQL Server 的登录名设置为 sa。



图 12-27　选择订阅服务器和订阅数据库

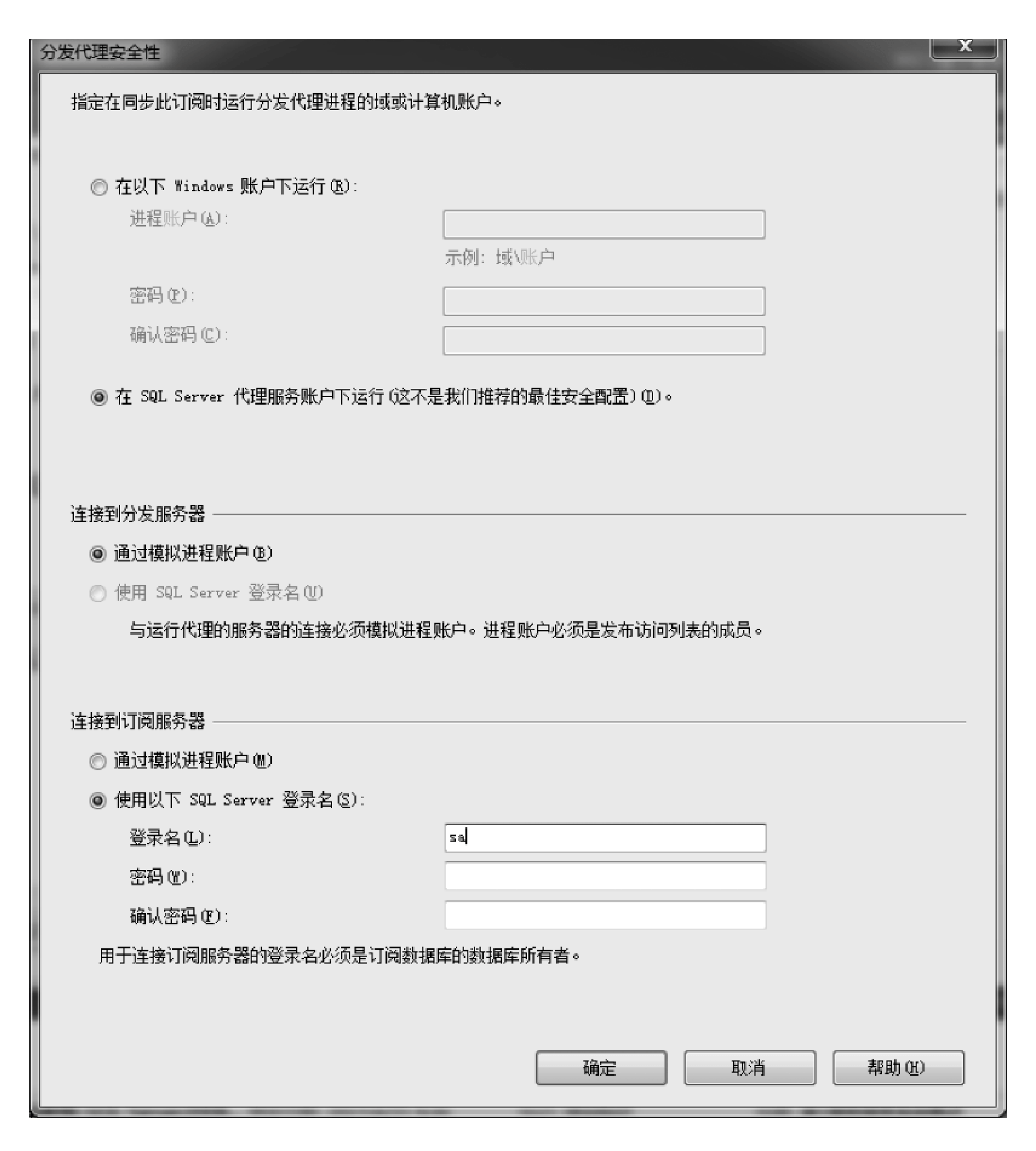


图 12-28　代理账户设置

(6) 单击"确定"按钮,分发代理安全性设置结果如图 12-29 所示。



图 12-29　分发代理安全性设置

(7) 单击"下一步"按钮,进入"同步计划"窗口,选择"连续运行",如图 12-30 所示。



图 12-30　设置同步计划

(8) 单击"下一步"按钮,进入"初始化订阅"窗口,选择"初始化时间"为"立即",如图 12-31 所示。



图 12-31 初始化订阅

(9) 单击"下一步"按钮,可以选择在向导结束时"创建订阅"或"生成包含创建订阅的步骤的脚本文件",这里只选择"创建订阅",如图 12-32 所示。



图 12-32 向导结束时创建订阅

(10) 单击“下一步”按钮，验证在向导中选择的选项，完成该向导，如图 12-33 所示。

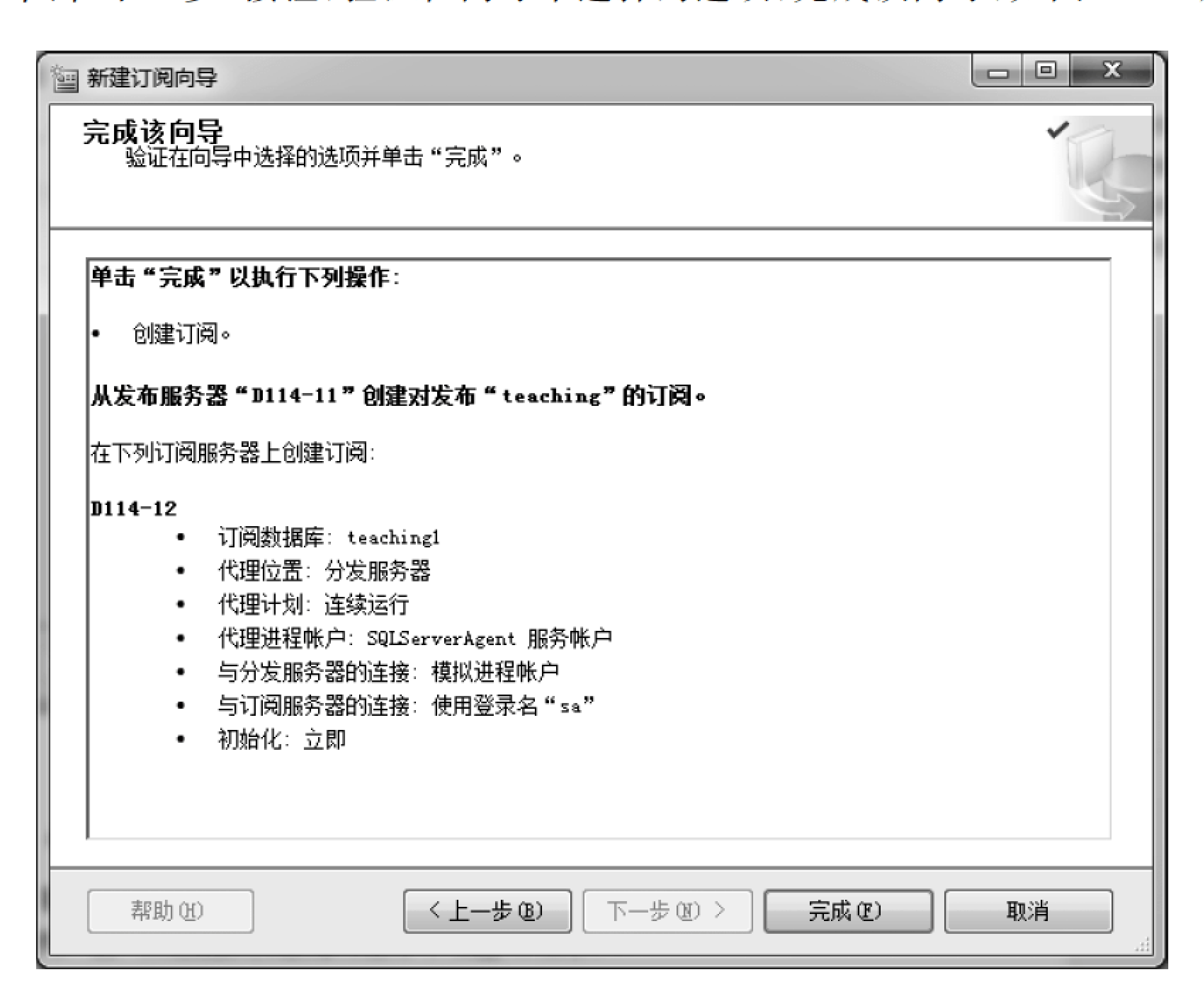


图 12-33　完成该向导

(11) 单击“完成”按钮，创建在 D114-12 上的订阅成功，如图 12-34 所示。



图 12-34　创建订阅成功

在 SQL Server 2008 服务器 D114-12 上创建订阅成功后，就可以在订阅数据库 teaching1 中看到 student 和 teacher 两个表。但快照复制是单向的，快照发布在订阅服务器收到已发布数据的初始快照后，对于发布服务器中对相关数据的修改，“快照代理”可以根据执行计划中设置的时间定时同步到订阅服务器中；反之，订阅服务器中对相关数据的修改，不能同步到发布服务器中。

4. 配置事务复制和合并复制

事务复制的配置过程与快照复制并没有太大的不同，只是在选择发布类型时，选择“事务发布”或“具有可更新订阅的事务发布”，如图 12-35 所示。

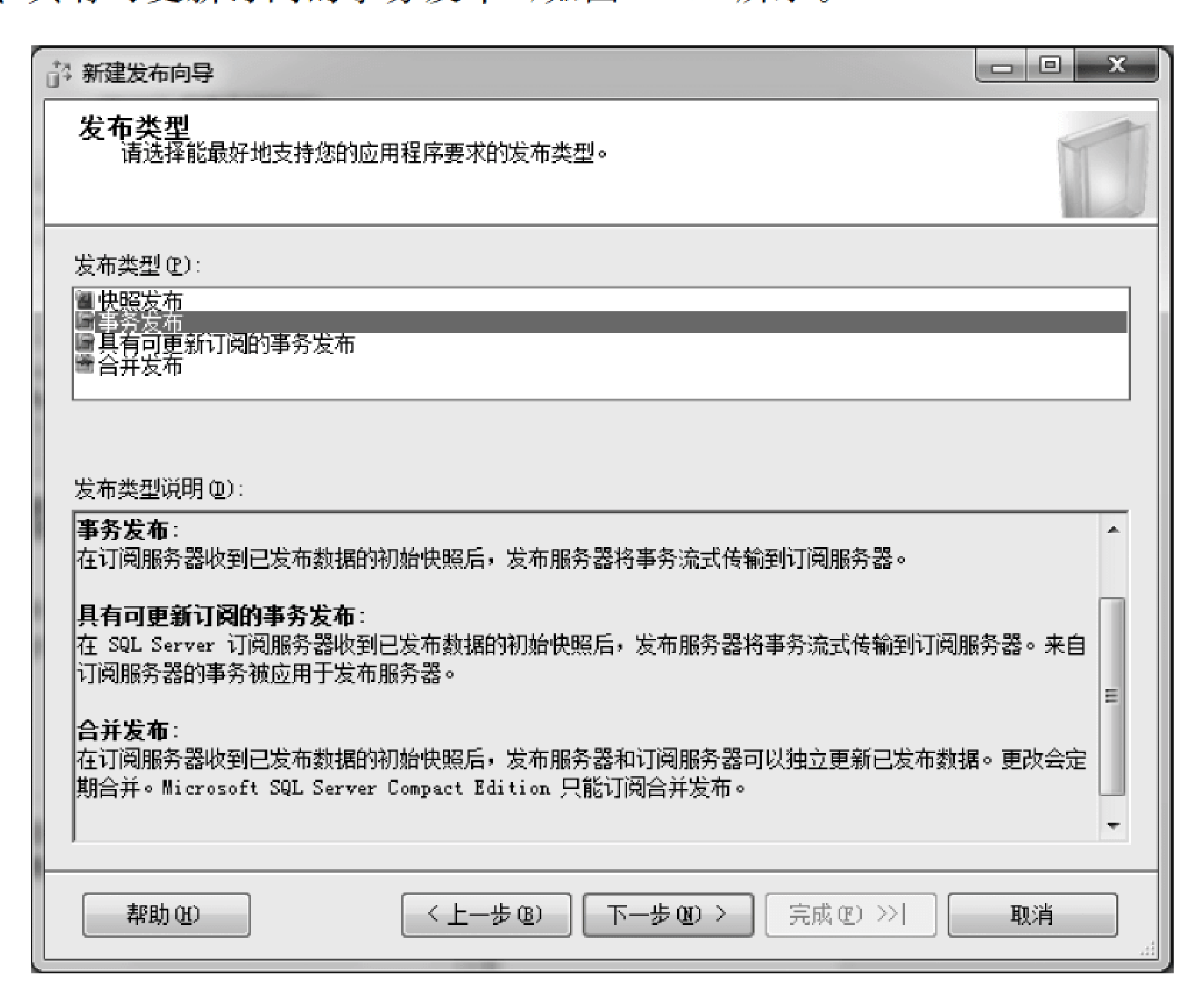


图 12-35　事务复制类型选择

二者的区别是：事务发布在订阅服务器收到已发布数据的初始快照后，发布服务器将事务按顺序传输到订阅服务器；而具有可更新订阅的事务发布除了上述功能，来自订阅服务器的事务也被应用于发布服务器。

另外，事务发布比快照发布多一个日志读取代理的安全设置，而具有可更新订阅的事务发布还要多一个队列读取器代理的安全设置。

合并复制的数据同步是双向的，其配置过程与快照发布的区别是：在选择发布类型时选择“合并发布”，如图 12-35 所示；单击“下一步”按钮指定订阅服务器类型，包括 SQL Server 2008、SQL Server 2005、SQL Server 2000 等多个版本都可以作为订阅服务器，在这里可以指定一个或多个，如图 12-36 所示。

另外，合并发布比快照发布多一个合并代理的安全设置。

总之，事务复制和合并复制的配置步骤与快照复制基本相同，这里不再讲述，读者可自行完成。



图 12-36　指定订阅服务器类型

习　题　12

1. 简述 bcp 导入导出的基本概念。

2. 将 bankcard 数据库中 4125428003351206000 账户的交易记录导出到 record1.txt 文件中。

3. 名词解释：项目、论文、发布者、订阅者、推订阅和拉订阅。

4. 简述什么是快照复制。

5. 简述什么是事务复制。

6. 简述什么是合并复制。

7. 简述 3 种复制类型的区别。

8. 准备两台 SQL Server 2008 服务器：一台作为发布和分发服务器；另一台作为订阅服务器，配置快照发布和订阅 bankcard 数据库中的 Trecord 表。

第三部分

数据库开发

第13章 关系数据库设计

关系数据库是用二维表结构模型表示数据的数据库，合理的数据库结构是数据库应用系统性能良好的基础和保证，但数据库的设计和开发却是一项庞大而复杂的工程。从事数据库设计的人员，不仅要掌握数据库知识和数据库设计技术，还要有程序开发的实际经验，了解软件工程的原理和方法；数据库设计人员必须深入应用环境，了解用户具体的专业业务；在数据库设计的前期和后期，与应用单位人员密切联系，共同开发，可大大提高数据库设计的成功率。

本章主要讲述关系数据库设计过程中的需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施、运行和维护等内容；并按照规范设计的方法和步骤介绍两个数据库设计实例。

13.1 数据库设计概述

数据库设计是根据用户需求设计数据库结构的过程。具体来说，数据库设计是对于给定的应用环境，在关系数据库理论的指导下，构造最优的数据库模式，在数据库管理系统上建立数据库及其应用系统，使之能有效地存储数据，满足用户的各种需求的过程。

数据库设计方法有多种，概括起来分为 4 类：直观设计法、规范设计法、计算机辅助设计法和自动化设计法。按照规范设计的方法，考虑数据库及其应用系统开发全过程，数据库设计可分为以下 6 阶段：需求分析阶段、概念结构设计阶段、逻辑结构设计阶段、物理结构设计阶段、数据库实施阶段以及数据库运行和维护阶段。

13.2 需求分析

需求分析是数据库设计的起点，需求分析就是数据库设计人员，通过仔细地调查和向用户详细地咨询，掌握用户的需求，理解用户的需求。需求分析的结果是否准确地反映了用户的实际要求，将直接影响到后面各个阶段的设计，并影响到设计结果是否合理和实用。如果投入大量的人力、物力、财力和时间开发出的软件却没人需要，那么所有的投入都是徒劳。如果费了很大的精力，开发一个软件，最后却不满足用户的要求，从而要重新开发，这种返工是让人痛心疾首的。

总之，需求分析为数据库的开发起到了决策的作用，提供了开发的方向，并指明了开发的策略，在数据库开发及维护中均起到了举足轻重的作用。可以说在一个大型数据库系统的开发中，它的作用要远远大于其他各个阶段。永远别忘了：数据库设计得合理、可行，能

够满足用户需求才是最重要的。

13.2.1 需求分析的任务

需求分析的任务是通过详细调查现实世界要处理的对象(组织、部门、企业等),充分了解原系统(手工系统或计算机系统)的工作概况,明确用户的各种需求,然后在此基础上确定新系统的功能。新系统必须充分考虑今后可能的扩充和改变,不能仅仅按当前应用需求来设计数据库。

调查的重点是数据和处理,通过调查、收集与分析,获得用户对数据库的如下要求。

1. 信息要求

信息要求指用户需要从数据库中获得信息的内容与性质。由信息要求可以导出数据要求,即在数据库中需要存储哪些数据。

2. 处理要求

处理要求指用户要完成什么处理功能,对处理的响应时间有什么要求,处理方式是批处理还是联机处理等。

3. 安全性与完整性要求

安全性要求是指对数据库的用户、角色、权限、加密方法等安全保密措施的要求。完整性要求是指对数据取值范围、数据之间各种联系的要求等等。

确定用户的最终需求往往是一件很困难的事,这是因为一方面用户缺少计算机知识,开始时无法确定计算机究竟能为自己做什么,不能做什么,往往不能准确地表达自己的需求,所提出的需求往往会不断变化;另一方面,设计人员缺少用户的专业知识,不易理解用户的真正需求,甚至误解用户的需求。因此,设计人员必须不断深入地与用户交流,才能逐步确定用户的实际需求。

13.2.2 需求分析的方法

进行需求分析首先是调查清楚用户的实际要求,与用户达成共识,然后分析与表达这些需求。

1. 调查用户需求的具体步骤

(1) 调查组织机构情况,包括了解该组织的部门组成情况、各部门的职责等,为分析信息流程做准备。

(2) 调查各部门的业务活动情况,包括了解各个部门输入和使用什么数据,如何加工处理这些数据,输出什么信息,输出到什么部门,输出结果的格式是什么,这是调查的重点。

(3) 在熟悉业务活动的基础上,协助用户明确对新系统的各种要求,包括信息要求、处理要求、完全性与完整性要求,这是调查的又一个重点。

(4) 确定新系统的边界,对前面调查的结果进行初步分析,确定哪些功能由计算机完成或将来准备让计算机完成,哪些活动由人工完成。由计算机完成的功能就是新系统应该实现的功能。在调查过程中,可以根据不同的问题和条件,使用不同的调查方法。

2. 常用的调查方法

(1) 跟班作业。通过亲身参加业务工作来了解业务活动的情况。这种方法可以比较准

确地理解用户的需求，但比较耗费时间。

(2) 开座谈会。通过与用户座谈来了解业务活动情况及用户需求。座谈时，参加者之间可以相互启发，一般可按职能部门组织座谈会。

(3) 询问或请专人介绍。一般应包括领导、管理人员、操作员等。

(4) 设计调查表请用户填写需求。如果调查表设计得合理，这种方法是很有效的，也易于被用户接受。

(5) 查阅记录。查阅与原系统有关的数据记录。

做需求调查时，往往需要同时采用上述多种方法。但无论使用何种调查方法，都必须有用户的积极参与和配合，最好能建立由双方人员参加的项目实施保障小组负责沟通联系。

13.3 概念结构设计

数据库概念结构设计是整个数据库设计的关键，此阶段要做的工作不是直接将需求分析得到的数据格式转换为 DBMS 能处理的数据模型，而是将需求分析得到的用户需求抽象为反映用户观点的概念模型。以此作为各种数据模型的共同基础，从而能更好、更准确地用某一 DBMS 实现这些需求。

描述概念结构的模型应具有以下几个特点：

(1) 有丰富的语义表达能力。能表达用户的各种需求，反映现实世界中各种数据及其复杂的联系，及用户对数据的处理要求等。

(2) 便于人的理解和交流。概念模型是系统分析师、数据库设计人员和用户之间的主要交流工具。

(3) 易于修改。概念模型能灵活地加以改变，以反映用户需求和环境的变化。

(4) 易于向各种数据模型转换。设计概念模型的最终目的是向某种 DBMS 支持的数据模型转换，建立数据库应用系统。

概念模型的表示方法很多，其中最为著名和使用最为广泛的是 P. P. Chen 于 1976 年提出的 E-R(Entity-Relationship，实体-联系)模型。

13.3.1 E-R 模型中的几个重要概念

1. 实体

客观存在并可以相互区分的事物叫实体(Entity)。从具体的人、物、事件到抽象的状态与概念都可以用实体抽象地表示；实体不仅可指事物本身，也可指事物之间的具体联系。例如，在学校里，一名学生、一名教师、一门课程、一次会议等都可称为实体。同类型实体的集合称为实体集(Entity Set)。例如，全体学生就是一个学生实体集。

2. 属性

属性(Attribute)是实体所具有的某些特性，通过属性对实体进行描述。实体是由属性组成的。一个实体本身具有许多属性，能够唯一标识实体的属性称为该实体的码。例如，学生实体可由学号、姓名、性别、年龄、系、专业等组成，(201602001、张强、男、19、计算机、软件

工程)这些属性组合起来就可以表示“张强”这个学生。

3. 码

一个实体往往有多个属性,这些属性之间是有关系的,它们构成该实体的属性集合。如果其中有一个属性或属性集能够唯一标识每一个实体,则称该属性或属性集为该实体的码(Key)。例如,学号是学生实体的码,每个学生都有一个属于自己的学号,通过学号可以唯一确定是哪位学生,在学校里,不可能有两个学生具有相同的学号。需要注意的是,实体的属性集可能有多个码,每一个码都称为候选码。但一个属性集只能确定其中一个候选码作为唯一标识。一旦选定,就称其为该实体的主码。

4. 联系

现实世界的事物之间是有联系的,即各实体型之间是有联系(Relationship)的。例如,教师实体与院系实体之间存在“属于”联系,学生和课程之间存在“选课”联系。实体间的联系是错综复杂的,但就两个实体型的联系来说,主要有以下 3 种情况。

(1) 一对一联系(1∶1):对于实体集 A 中的每一个实体,实体集 B 中至多有一个实体与之对应,反之亦然,则称实体集 A 与实体集 B 具有一对一联系,记为 1∶1,如图 13-1 所示。例如,部门与经理之间联系,学校与校长之间的联系等等就是一对一联系。

(2) 一对多联系(1∶M):对于实体集 A 中的每一个实体,实体集 B 中有多个实体与之对应;反过来,对于实体集 B 中的每一个实体,实体集 A 中至多有一个实体与之对应,则称实体集 A 与实体集 B 具有一对多联系,记为 1∶M,如图 13-2 所示。例如,一个班可以有多个学生,但一个学生只能属于一个班。班级与学生之间的联系就是一对多联系。

(3) 多对多联系(M∶N):对于实体集 A 中的每一个实体,实体集 B 中有多个实体与之对应;反过来,对于实体集 B 中的每一个实体,实体集 A 中也有多个实体与之对应,则称实体集 A 与实体集 B 具有多对多联系,记为 M∶N,如图 13-3 所示。

例如,学生在选课时,一个学生可以选多门课程,一门课程也可以被多个学生选,则学生和课程之间具有多对多联系。

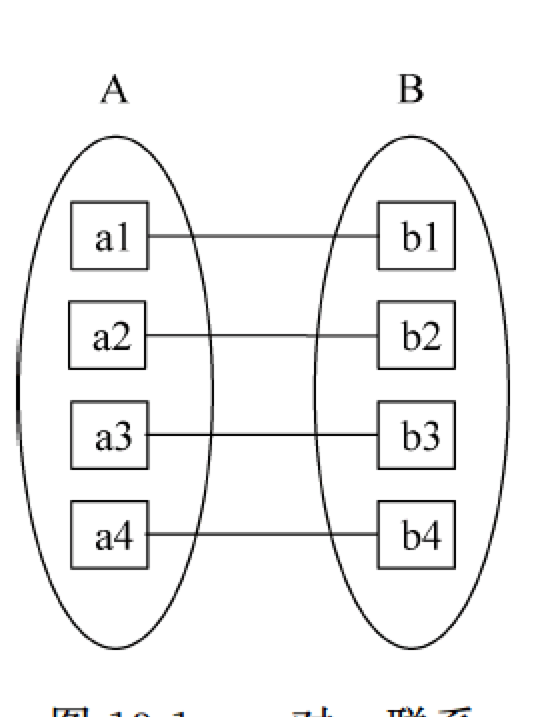


图 13-1 一对一联系

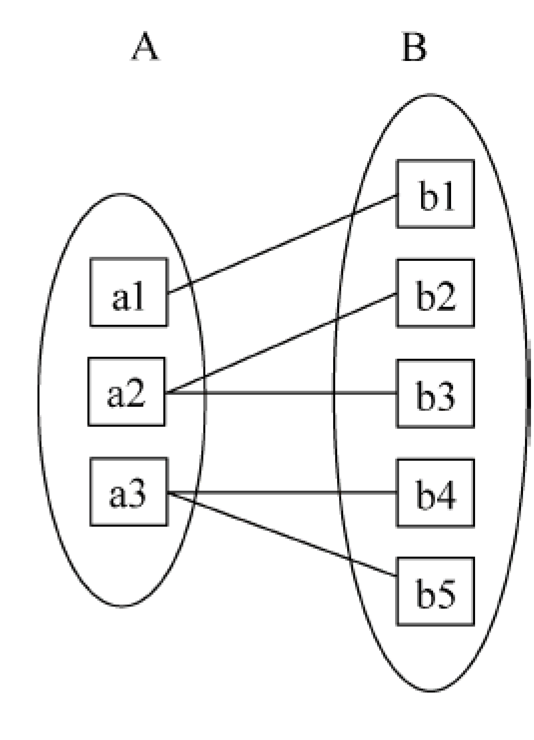


图 13-2 一对多联系

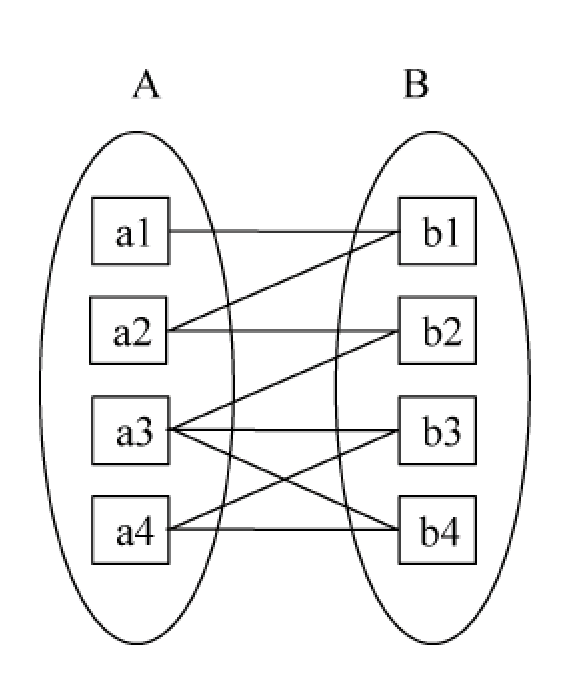


图 13-3 多对多联系

13.3.2 E-R 模型的通用表示形式

E-R 模型的图形表示称为 E-R 图,其通用表示形式如下:

(1) 用矩形表示实体,在矩形框内写上实体名。例如,学生实体、班级实体,如图 13-4

所示。

(2) 用椭圆形表示实体的属性,并用无向边把实体和属性连接起来。例如,学生实体有学号、姓名、性别、出生日期属性,班级实体有班级名、班主任属性,如图 13-5 所示。

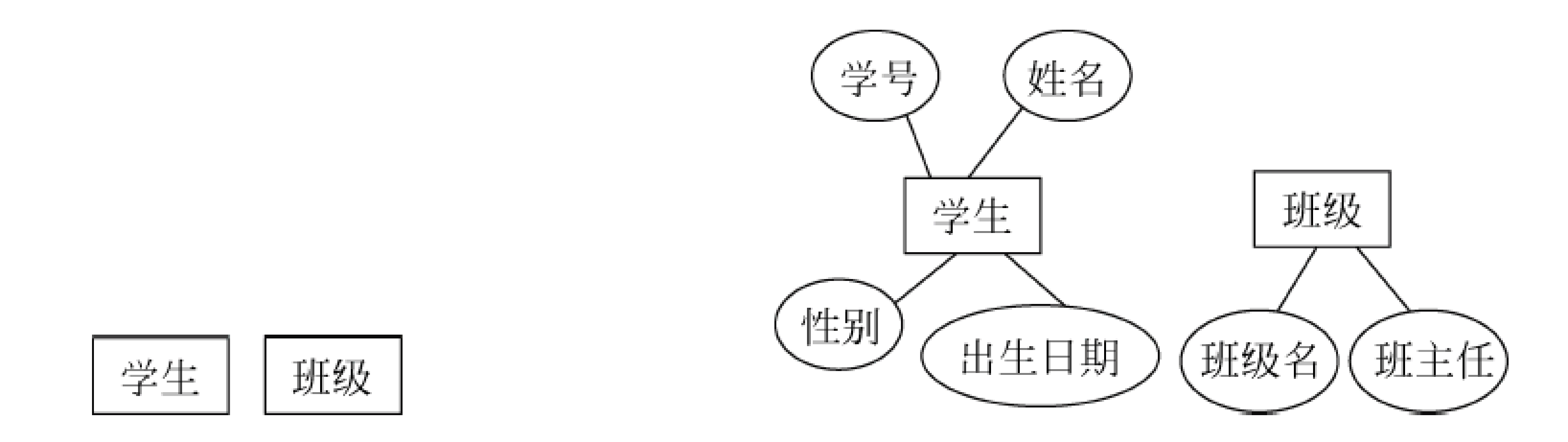


图 13-4　学生、班级实体　　　图 13-5　学生、班级实体及属性

(3) 用菱形表示实体间的联系,在菱形框内写上联系名,用无向边分别把菱形框与有关实体连接起来,在无向边旁注明联系的类型。如果实体间的联系也有属性,则把属性和菱形框也用无向边连接起来。例如,学生实体和班级实体的联系,如图 13-6 所示。

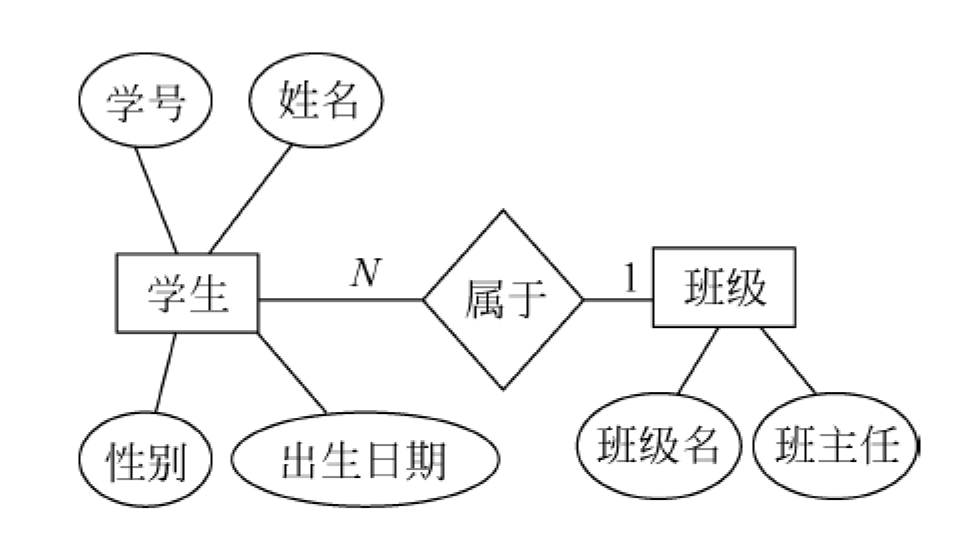


图 13-6　学生实体和班级实体间的联系

【例 13-1】 有一个简单的学生信息数据库系统,包含班级、学生和课程实体,其中一个班可以有若干个学生,一个学生只能属于一个班;一个学生可以选修多门课,一门课也可以有多个学生选修,学生选课后有成绩。该数据库系统的 E-R 模型如图 13-7 所示。

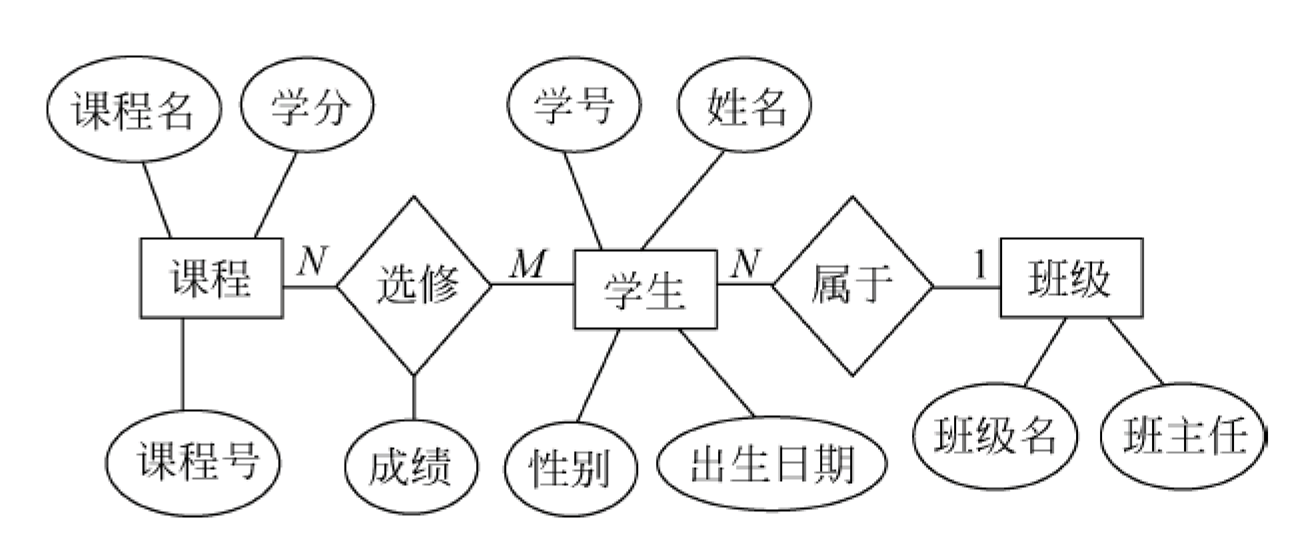


图 13-7　学生信息数据库系统的 E-R 图

【例 13-2】 有一个高等学校信息数据库系统,包含学生、教师、专业、教科书和课程 5 个实体,其中一个专业可以有若干个学生,一个学生只能属于一个专业;一个专业可以开多门课,一门课只能在一个专业开课;一个专业可以有若干个教师,一个教师只能属于一个专业;一位教师可以讲授多门课,一门课也可以有多位教师讲授,每个教师讲授每门课都有一个开课学期;一个专业可以订购若干本教科书,一本教科书也可以有多个专业订购,每个专业订购的每本教科书都有一个数量;一个学生可以选修多门课,每一门课都可以有多个学生选修,学生选课后有成绩。该数据库系统的 E-R 模型如图 13-8 所示。

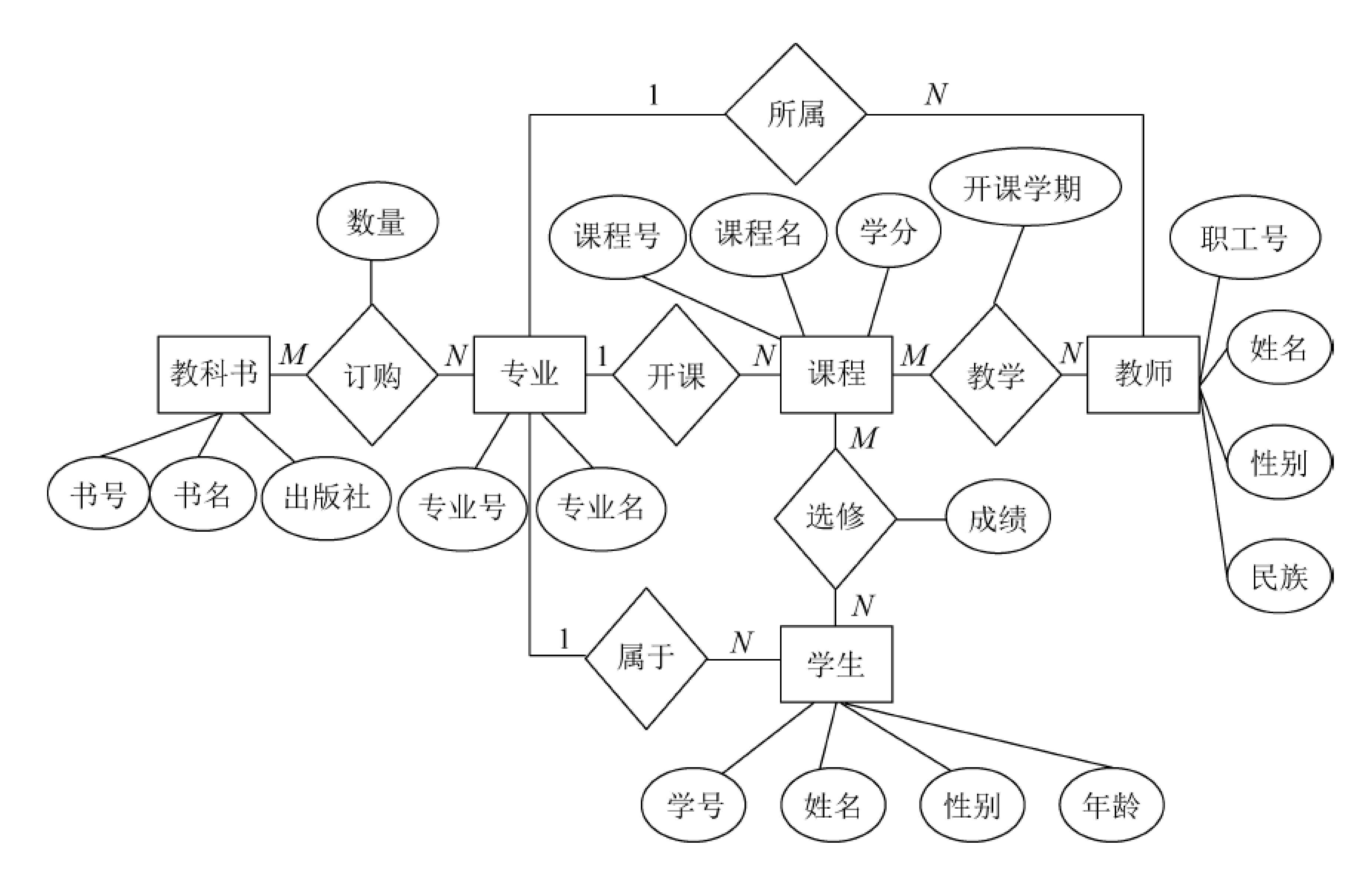


图 13-8 高等学校信息数据库系统的 E-R 图

13.3.3 概念结构设计的步骤

人们归纳出设计概念模型的策略很多,其中最常用的策略是自底向上法。自底向上法可以分为两步:

(1) 进行数据抽象,设计局部概念模型,即设计局部 E-R 图;

(2) 集成各局部 E-R 图,形成全局 E-R 图,即 E-R 图的集成。

1. 设计局部 E-R 图

局部 E-R 图的设计步骤包括如下 4 步:

1) 确定局部 E-R 图描述的范围

根据需求分析所产生的文档,可以确定每个局部 E-R 图描述的范围。通常采用的方法是将功能划分为几个系统,每个系统又可分为几个子系统。设计局部 E-R 模型的第一步就是划分适当的系统或子系统,在划分时过细或过粗都不太合适。划分过细将造成大量的数据冗余和不一致,过粗有可能漏掉某些实体。

一般可以遵循以下两条原则进行功能划分:

- 独立性原则。划分在一个范围内的应用功能具有独立性与完整性,与其他范围内的应用有最少的联系。
- 规模适度原则。局部 E-R 图规模应适度,一般以 6 个左右实体为宜。

2) 确定局部 E-R 图的实体

根据需求分析说明书,将用户的数据需求和处理需求中涉及的数据对象进行归类,指明对象的身份,是实体、联系还是属性。

3) 定义实体的属性

根据上一步确定的实体的描述信息来确定其属性。

4) 定义实体间的联系

确定了实体及其属性后,就可以定义实体间的联系了。实体间的联系按其特点可分为

3 种：存在性联系(如学生有所属的班级)、功能性联系(如教师要教学生)、事件性联系(如学生借阅书籍)。实体间的联系方式分为一对一、一对多、多对多 3 种。

设计完成某一局部结构的 E-R 图后，再看还有没有其他的局部 E-R 图的实体，如果有则转到第 2)步继续，直到所有的局部 E-R 图都设计完为止。

2. 局部 E-R 图的集成

由于局部 E-R 图反映的只是单位局部子功能对应的数据视图，可能存在不一致的地方，还不能作为逻辑设计的依据，这时可以去掉不一致和重复的地方，将各个局部 E-R 图合并为全局 E-R 图，即局部 E-R 图的集成。

一般说来，局部 E-R 图的集成可以有两种方式：第一种是多个分 E-R 图一次集成；第二种是逐步集成，用累加的方式一次集成两个分 E-R 图。第一种方式比较复杂，做起来难度较大。第二种方式每次只集成两个分 E-R 图，可以降低复杂度。

无论采用哪种集成法，每一次集成都分为两个阶段：第一步是合并，以消除各局部 E-R 图之间的不一致情况，生成初步全局 E-R 图；第二步是优化，消除不必要的数据冗余，包括冗余的数据和实体间冗余的联系，生成最终全局 E-R 图。

【例 13-3】 在如图 13-9 所示的简易教学信息数据库系统 E-R 图中，学生和教师之间不需要直接的联系，因为不常用，偶尔在用户程序中需要用到二者之间的联接查询时，可以通过课程实体间接实现，所以学生和教师之间为多余的联系；同样学生和学院之间也为多余的联系；平均成绩为多余的属性，因为它完全可以通过其他属性计算得到，除非应用程序有特殊需要，否则不需要设置此属性。

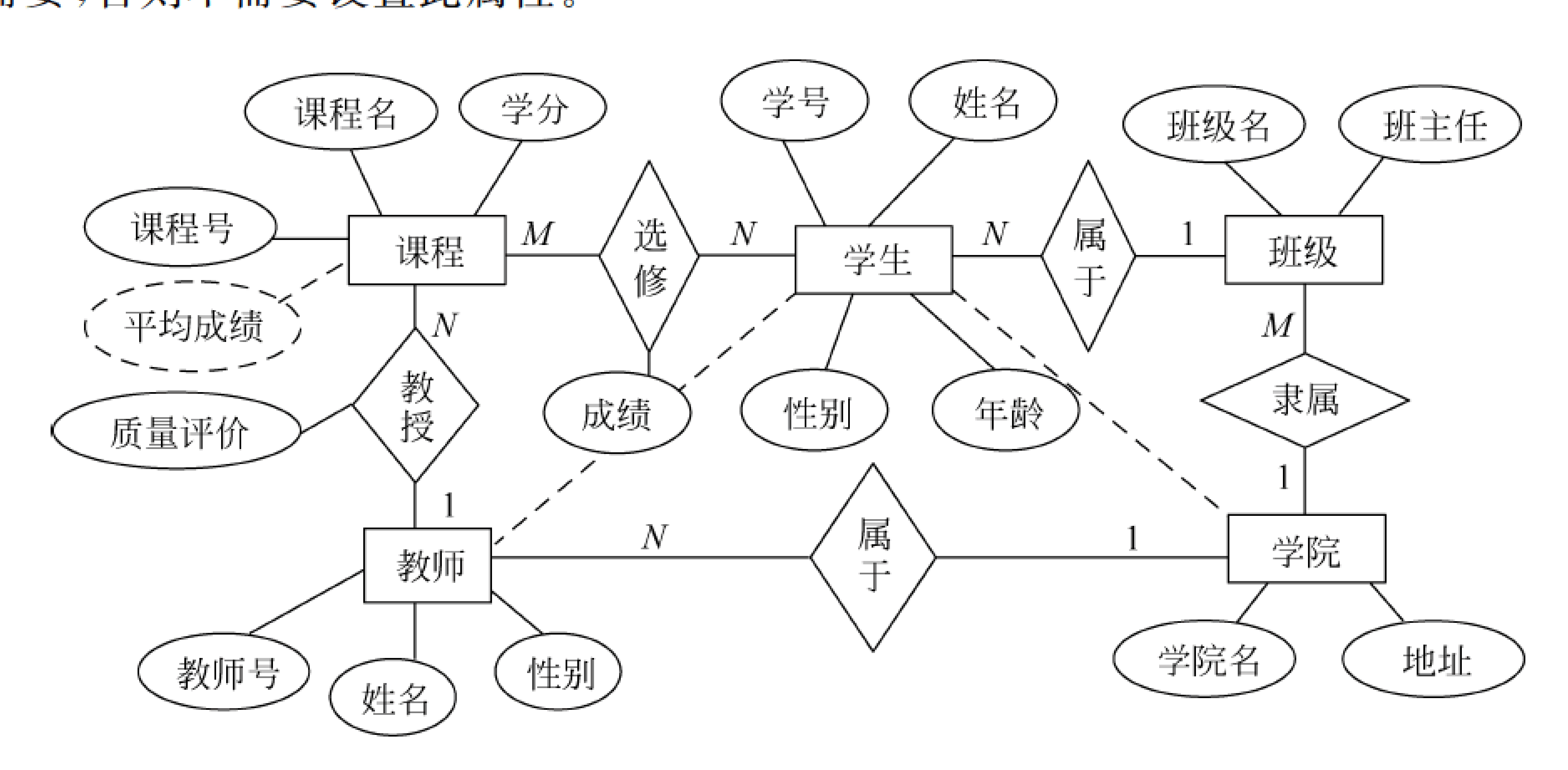


图 13-9　简易教学信息数据库系统 E-R 图

13.4　逻辑结构设计

数据库概念设计阶段得到的数据模式是用户需求的形式化，它独立于具体的计算机系统和 DBMS。为了建立用户所要求的数据库，必须把上述数据模式转换成某个具体的 DBMS 所支持的数据模式，并以此为基础建立相应的外模式，这是数据库逻辑设计的任务，是数据库结构设计的重要阶段。

逻辑设计的主要目标是产生一个 DBMS 可处理的数据模型，该模型必须满足数据库的

存取、一致性及运行等各方面的用户需求。逻辑结构设计阶段一般要分为 3 步进行：将 E-R 图转化为关系数据模型；关系模式的优化；设计用户视图。

13.4.1 将 E-R 图转化为关系数据模型

关系数据模型是一组关系模式的集合，而 E-R 图是由实体、属性和实体之间的联系三要素组成的。所以，将 E-R 图转化为关系数据模型实际上是要将实体、属性和实体之间的联系转化为关系模式。

转化过程中要遵循如下原则。

1. 实体的转换

一个实体转化为一个关系模式，实体的属性就是该关系模式的属性，实体的主码就是该关系模式的主码。

2. 联系的转换

(1) 两实体集间 1∶1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。

① 转换为一个独立的关系模式。

转换后的关系模式中关系的属性包括与该联系相连的各实体的主码以及联系本身的属性（如果有），关系的主码为两个实体的主码的组合。

【例 13-4】 将如图 13-10 所示的 E-R 图按方法①转换为关系模式。

首先，将其中的两个实体按实体的转换原则转换成两个关系模式：班级和班长；然后，将班级和班长之间的一对一联系转换为一个独立的关系模式：班级-班长。3 个关系模式如下：

班级(<u>班号</u>,系别,班主任,入学时间)
班长(<u>学号</u>,姓名,性别,年龄)
班级－班长(<u>班号,学号</u>,任期)

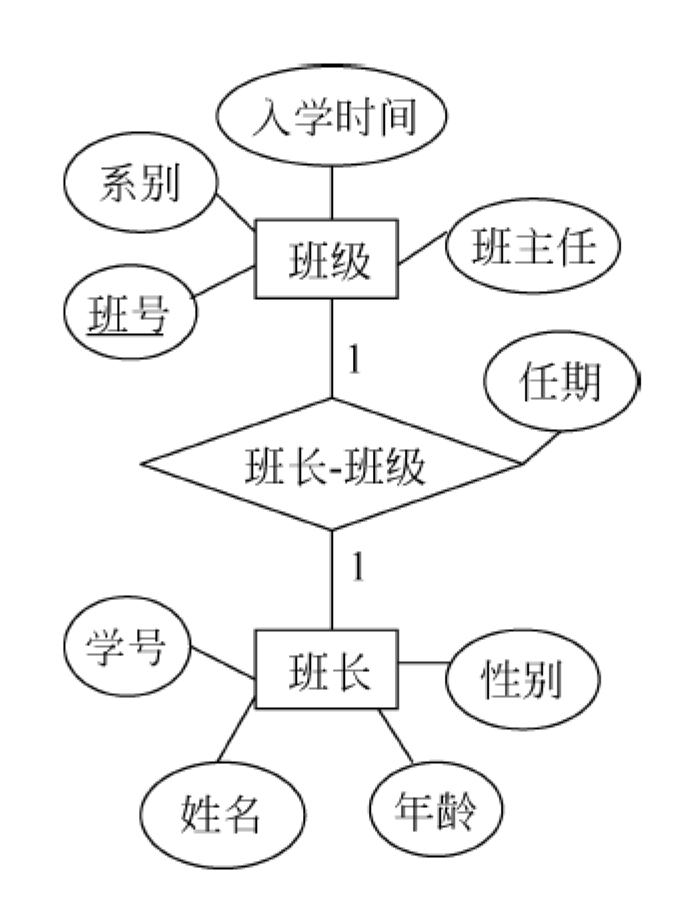


图 13-10　1∶1 联系的 E-R 图

② 与某一端对应的关系模式合并。

合并后关系模式的属性包括自身关系模式的属性和另一关系模式的主码及联系本身的属性，合并后关系的主码不变。

【例 13-5】 将如图 13-10 所示的 E-R 图按方法②转换为关系模式。

首先将其中的两个实体按实体的转换原则转换成两个关系模式：班级和班长；然后，将班级和班长之间的一对一联系合并到班长或班级实体中。两个关系模式如下：

班级(<u>班号</u>,系别,班主任,入学时间)
班长(<u>学号</u>,姓名,性别,年龄,班号,任期)

或

班级(<u>班号</u>,系别,班主任,入学时间,班长学号,班长任期)
班长(<u>学号</u>,姓名,性别,年龄)

(2) 两实体集间 1∶N 联系可以转换为一个独立的关系模式,也可以与 N 端对应的关系模式合并。

① 转换为一个独立的关系模式。

关系的属性包括与该联系相连的各实体的主码以及联系本身的属性,关系的主码为 N 端实体的主码。

【例 13-6】 将如图 3-11 所示的 E-R 图按方法①转换为关系模式。

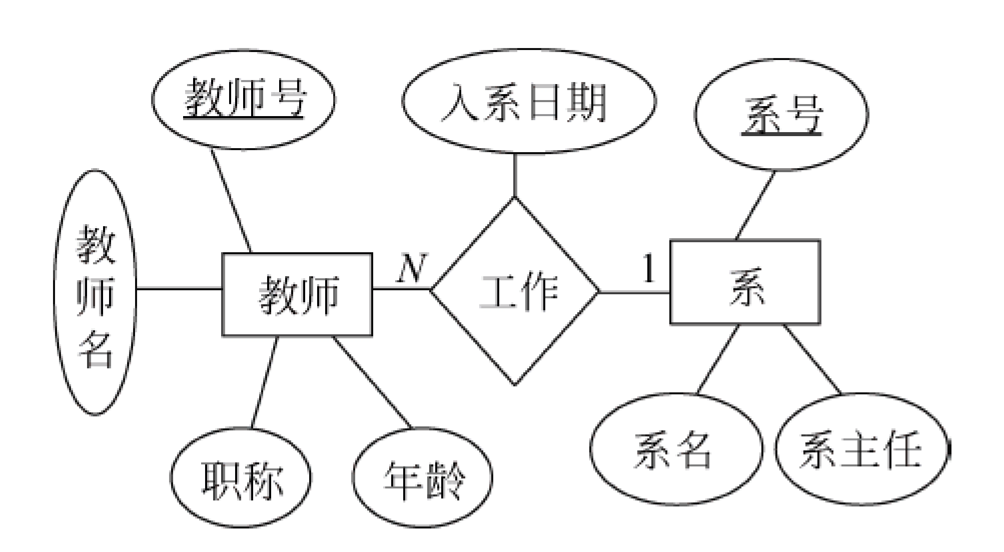


图 13-11　1∶N 联系的 E-R 图

首先将其中的两个实体按实体的转换原则转换成两个关系模式:系和教师;然后,将系和教师之间的一对多联系转换为一个独立的关系模式:工作。3 个关系模式如下:

系(<u>系号</u>,系名,系主任)
教师(<u>教师号</u>,教师名,年龄,职称)
工作(<u>教师号</u>,系号,入系日期)

② 与 N 端对应的关系模式合并。

合并后关系的属性包括在 N 端关系中加入 1 端关系的主码和联系本身的属性,合并后关系的主码不变。

【例 13-7】 将如图 13-11 所示的 E-R 图按方法②转换为关系模式。

首先将其中的两个实体按实体的转换原则转换成两个关系模式:系和教师;然后,将系和教师之间的一对多联系合并到多端教师实体中。两个关系模式如下:

系(<u>系号</u>,系名,系主任)
教师(<u>教师号</u>,教师名,年龄,职称,系号,入系日期)

注意: 实际使用中两实体间 1∶1 和 1∶N 联系通常采用方法②,以减少数据库系统中的关系模式,因为多一个关系模式就意味着查询过程中可能要进行连接运算,而连接运算会大大降低查询的效率。

(3) 同一实体集内实体间的 1∶N 联系,可在这个实体集所对应的关系模式中多设一个属性,用来作为与该实体相联系的另一个实体的主码。

比如学生实体集中有部分学生是班长,就可以在学生关系模式中加入一个"班长学号"属性,表示这个学生是所在班的班长。

(4) 两实体集间 M∶N 联系,必须为联系产生一个新的关系模式。该关系模式中至少包含被它所联系的双方实体的主码,若联系中有属性,也要并入该关系模式中。如果没有指定另外的属性(比如此联系的 ID)作为该关系的主码,则该关系的主码为双方实体的主码的组合。

【例 13-8】 将如图 13-12 所示的 E-R 图转换成对应的关系模式。

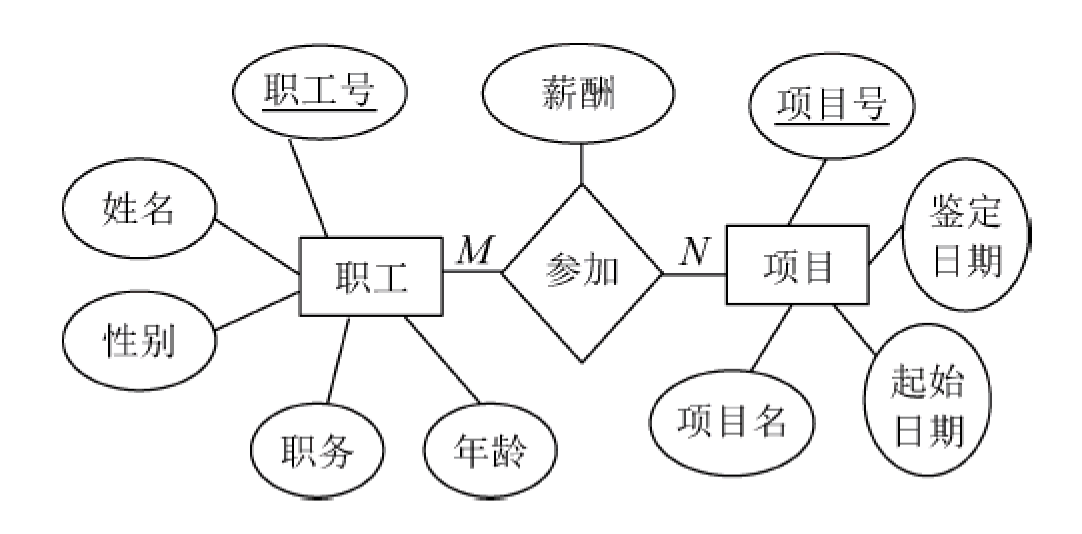


图 13-12　两实体间 $M:N$ 联系的 E-R 图

首先，将其中的两个实体按实体的转换原则转换成两个关系模式：职工和项目；然后，将职工和项目之间的多对多联系转换为一个独立的关系模式：参加。3 个关系模式如下：

职工(<u>职工号</u>,姓名,性别,年龄,职务)
项目(<u>项目号</u>,项目名,起始日期,鉴定日期)
参加(<u>职工号,项目号</u>,薪酬)

(5) 同一实体集内实体间 $M:N$ 联系，必须为联系产生一个新的关系模式。该关系模式中至少包含被它所联系的双方实体的主码，若联系有属性，也要并入该关系模式中。

比如学生实体集中有部分学生是社团的负责人，而一个学生可以参加多个社团，一个社团也可以有多个学生，所以学生实体集中学生和社团负责人就存在 $M:N$ 联系。学生实体集转换为一个学生关系，学生实体集内实体间 $M:N$ 联系就生成一个新的关系模式，该关系中至少包含"学号"和"社团负责人编号"两个属性。

(6) 两个以上实体集之间 $M:N$ 的联系，必须为联系产生一个新的关系模式，该关系模式中至少包含被它所联系的所有实体的主码，若联系有属性，也要并入该关系模式中，关系的主码可以指定一个单独的属性(比如此联系的 ID)，否则为它所联系的所有实体的主码的组合。

13.4.2　关系模式的优化

通常情况下，数据库逻辑设计的结果不是唯一的。为了进一步提高数据库应用系统的性能，还应努力减少关系模式中存在的各种异常，改善完整性、一致性和存储效率。为了提高数据库应用系统的性能，规范化后的关系模式还需要进行修改，调整结构，这就是关系模式的进一步优化，通常采用合并或分解的方法。

关系模式的优化方法如下：

(1) 确定函数依赖。

(2) 对于各个关系模式之间的函数依赖进行极小化处理，消除冗余的函数依赖。

(3) 按照函数依赖的理论对关系模式逐一进行分析，考查是否存在部分函数依赖、传递函数依赖等，确定各关系模式分别属于第几范式，将不符合规范的关系模式规范化。

(4) 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

(5) 对关系模式进行必要的合并或分解。

规范化理论为数据库设计人员判断关系模式优劣提供了理论标准，可用来预测模式可

能出现的问题，使数据库设计工作有了严格的理论基础。

1. 函数依赖的概念

实体的联系有两类：一类是实体与实体之间的联系；另一类是实体内部各属性间的联系。定义属性值间的相互关联（主要体现在值的相等与否），这就是数据依赖，它是数据库模式设计的关键。

所谓函数依赖，是指在关系 R 中，X、Y 为 R 的两个属性或属性组，如果关系 R 存在：对于 X 的每一个具体值，Y 都只有一个具体值与之对应，则称属性 Y 函数依赖于属性 X。记作 $X \to Y$。当 Y 不函数依赖于 X 时，记作：$X \nrightarrow Y$。当 $X \to Y$ 且 $Y \to X$ 时，则记作：$X \leftrightarrow Y$。

简单表述：如果属性 X 的值决定属性 Y 的值，那么属性 Y 函数依赖于属性 X；或者，如果知道 X 的值，就可以获得 Y 的值。

【例 13-9】 学生情况表，如表 13-1 所示，其对应的关系模式可描述为：学生情况（学号，姓名，专业名，性别，出生日期，总学分）。其中，学号为关键字，求其函数依赖关系有哪些。

表 13-1　学生情况表

学号	姓名	专业名	性别	出生日期	总学分
20151101	王　林	计算机	男	1996-02-10	150
20151102	程大伟	计算机	男	1997-02-01	150
20151103	王燕妮	计算机	女	1995-10-06	150
20152104	张　明	网络	男	1997-08-26	150
20152106	李方方	网络	女	1996-11-20	150

由函数依赖的定义可知，存在如下的函数依赖关系集：

学号→姓名；学号→专业名；学号→性别；学号→出生日期；学号→总学分

(1) 非平凡函数依赖和平凡函数依赖。

设关系模式 $R(U)$，U 是 R 上的属性集，X、$Y \subseteq U$；如果 $X \to Y$，且 $Y \subseteq X$，则称 $X \to Y$ 为平凡的函数依赖；如果 $X \to Y$，且 Y 不是 X 的子集，则称 $X \to Y$ 为非平凡的函数依赖。

【例 13-10】 在学生课程（学号，课程号，成绩）关系中，存在函数依赖为：（学号，课程号）→成绩，为非平凡的函数依赖；而（学号，课程号）→学号，为平凡的函数依赖。

以下所讨论的全部为非平凡的函数依赖。

(2) 完全函数依赖和部分函数依赖。

设关系模式 $R(U)$，U 是 R 上的属性集，X、$Y \subseteq U$；如果 $X \to Y$，并且对于 X 的任何一个真子集 Z，$Z \to Y$ 都不成立，则称 Y 完全函数依赖于 X；如果 $X \to Y$，但对于 X 的某一个真子集 Z，有 $Z \to Y$ 成立，则称 Y 部分函数依赖于 X。

【例 13-11】 在学生课程（学号，课程号，成绩）关系中，"学号，课程号"是主码，由于"学号→成绩"不成立，"课程号→成绩"也不成立，因此，"成绩"完全函数依赖于（学号，课程号）。

【例 13-12】 在学生课程（学号，姓名，课程号，成绩）关系中，"学号，课程号"是主码，由于"学号→姓名"成立，因此，"姓名"部分函数依赖于（学号，课程号）。

(3) 传递函数依赖。

设关系模式 $R(U)$，$X \subseteq U$，$Y \subseteq U$，$Z \subseteq U$；如果 $X \to Y$，$Y \nrightarrow X$，且 $Y \to Z$ 成立，则称 $X \to Z$

为传递函数依赖。

【例 13-13】 学生关系(学号,姓名,性别,年龄,所在系,系主任),“学号”为主码,其上的函数依赖包括:学号→姓名,学号→性别,学号→年龄,学号→所在系,所在系→系主任,则学号→系主任为传递函数依赖。

2. 关系模式规范化

若设计一个描述学校的数据库:一个系有若干学生,一个学生只属于一个系;一个系只有一名主任;一个学生可以选修多门课程,每门课程有若干学生选修;每个学生所学的每门课程都有一个成绩,如表 13-2 所示。

表 13-2　学生信息表

学号	姓名	年龄	系别	系主任	课程号	成绩
20161001	赵　红	20	计算机	张力	C1	90
20161001	赵　红	20	计算机	张力	C2	85
20162002	王小明	17	数学	王晓	C5	57
20162002	王小明	17	数学	王晓	C6	80
20162002	王小明	17	数学	王晓	C7	76
20162002	王小明	17	数学	王晓	C4	70
20163003	吴　林	19	信息	赵钢	C1	75
20163003	吴　林	19	信息	赵钢	C2	70
20161004	张　涛	21	计算机	张力	C1	93

则上述数据库对应的关系模式为:学生信息表(学号,姓名,年龄,系别,系主任,课程号,成绩),(学号,课程号)为主码。

上述关系模式中存在以下问题:

① 数据冗余。数据在数据库中的重复存放称为数据冗余。冗余度大,不仅浪费存储空间,重要的是在对数据进行修改时,易造成数据的不一致性。例如,系名、学生姓名、年龄等等都要重复存储多次,当它们发生改变时,就需要修改多次,一旦遗漏就会使数据不一致。

② 更新异常。因为存在数据冗余,所以更新数据时,维护数据完整性代价就会增大。如果某学生改名,则该学生的所有记录都要逐一修改姓名的值;稍有不慎,就有可能漏改某些记录。

③ 插入异常。无法插入某部分信息称为插入异常,即该插的数据插不进去。例如,如果一个系刚成立,尚无学生,我们就无法把这个系及其系主任的信息存入数据库。因为学号与课程号是主码,主码不能为空。

④ 删除异常。不该删除的数据不得不删除。例如,如果某个系的学生全部毕业了,那么在删除该系学生信息的同时,把这个系及其系主任的信息也丢掉了。

上述关系模式设计不合理,不是一个好的关系模式,“好”的关系模式不会发生插入异常、删除异常、更新异常,数据冗余也应尽可能少。

关系模式规范化的目的就是解决关系模式中存在的数据冗余、插入和删除异常以及更新异常等问题。其基本思想是消除数据依赖中的不合适部分,使各关系模式达到某种程度的分离,使一个关系描述一个概念、一个实体或实体间的一种联系。因此,规范化的实质是概念的单一化。

关系数据库中的关系必须满足一定的规范化要求，对于不同的规范化程度可用范式来衡量。范式(Normal Form)是符合某一种级别的关系模式的集合，是衡量关系模式规范化程度的标准，达到的关系才是规范化的。目前主要有6种范式：第一范式、第二范式、第三范式、BC范式、第四范式和第五范式。满足最低要求的叫第一范式，简称为1NF。在第一范式基础上进一步满足一些要求的为第二范式，简称为2NF。其余以此类推。显然各种范式之间存在联系：1NF⊆2NF⊆3NF⊆BCNF⊆4NF⊆5NF。

通常把某一关系模式 R 为第 n 范式简记为 $R \subseteq n$NF。

范式的概念最早是由E. F. Codd提出的。在1971—1972年，他先后提出了1NF、2NF、3NF的概念，1974年他又和Boyee共同提出了BCNF的概念，1976年Fagin提出了4NF的概念，后来又有人提出了5NF的概念。在这些范式中，最重要的是3NF和BCNF，它们是进行规范化的主要目标。

一个低一级范式的关系模式通过模式分解可以转换为若干个高一级范式的关系模式的集合，这个过程称为规范化。通常在实际情况下，规范化到3NF就可以了。

1）第一范式

设 R 是一个关系模式，如果 R 的每个属性的值域都是不可分的简单数据项(原子值)的集合，则称这个关系模式属于第一范式，简记作 $R \in 1$NF。

也可以说，如果关系模式 R 的每一个属性都是不可分解的，则 R 为第一范式的模式，1NF是规范化最低的范式。

在任何一个关系数据库系统中，关系至少应该是第一范式，不满足第一范式的数据库模式不能称为关系数据库。但注意，第一范式不能排除数据冗余和异常情况的发生。

例如，表13-3描述的是某单位职工情况。

表13-3　职工情况表

职工号	姓名	工　　资		
		基本工资	职务工资	工龄工资
20011	李　岚	3290	1200	1430
20012	王晓江	3000	1300	1240
20013	张　华	3800	1500	1620

由于表13-3中工资一项包括3部分，不满足每个属性不能分解的要求，是非规范化表，不是第一范式。可规范化为表13-4。

表13-4　规范化后的职工情况表

职工号	姓名	基本工资	职务工资	工龄工资
20011	李　岚	3290	800	430
20012	王晓江	3000	800	340
20013	张　华	3800	1500	1620

2）第二范式

如果关系模式 R 属于第一范式，且它的每个非主属性都完全函数依赖于码(候选码)，则称 R 为满足第二范式的关系模式，简记作 $R \in 2$NF。

注意：在一个关系中，包含在任何候选码中的各个属性称为主属性；不包含在任何候选码中的属性称为非主属性。

从第二范式开始，规范化时，我们采用的是每个关系的最小函数依赖集，最小函数依赖集是符合以下条件的函数依赖集 F：

① F 中任何一个函数依赖的右部仅含有一个属性。

② F 中的所有函数依赖的左边都没有冗余属性。

③ F 中不存在冗余的函数依赖。

【例 13-14】 学生关系 S(学号，姓名，性别，课程号，成绩)，其中学号和课程号的组合为主码，姓名、性别、学分为非主属性，关系 S 中的最小函数依赖集为：

学号→姓名，学号→性别，(学号，课程号)→成绩。

实际上，函数依赖"(学号，课程号)→姓名"也成立，但左边的"课程号"是多余的；函数依赖"学号→学号"也成立，但这是一个冗余的函数依赖。

两个推论：

① 关系 $R \in$ 1NF，且其主关键字只有一个属性，则关系 R 一定属于第二范式。

【例 13-15】 在关系 R(学号，姓名，性别，出生日期)中主码为学号，姓名、性别、出生日期为非主属性，存在下列最小函数依赖集：

学号→姓名，学号→性别，学号→出生日期

由于每个非主属性都完全函数依赖于码，所以该关系 $R \in$ 2NF。

② 主关键字是属性的组合，这样的关系模式可能不属于第二范式。

对于例 13-14 中的最小函数依赖集，存在非主属性(姓名和性别)部分函数依赖于码，故关系 S 不属于 2NF。对上述关系模式进行分解，分解方法：每个非主属性与它所依赖的属性组成新关系，新关系要尽可能的少，新关系的主码为函数依赖的左侧属性或属性集。则上述关系模式分解为两个关系：

$S1$(学号，姓名，性别)和 $S2$(学号，课程号，学分)；且 $S1 \in$ 2NF，$S2 \in$ 2NF。

【例 13-16】 职工信息关系 P(职工号，姓名，职称，项目号，项目名称，项目排名)，其中，主码为(职工号，项目号)，非主属性为(姓名，职称，项目名称，项目排名)关系 P 中的最小函数依赖集如下：

职工号→姓名，职工号→职称，项目号→项目名称，(职工号，项目号)→项目排名

由于存在非主属性部分依赖于码，故关系 P 不属于 2NF。对上述关系模式进行分解，分解为 3 个关系：职工信息表(职工号，姓名，职称)、项目排名表(职工号，项目号，项目排名)和项目表(项目号，项目名称)。

3) 第三范式

如果关系模式 R 属于第二范式，且没有一个非主属性传递函数依赖于码，则称 R 为满足第三范式的关系模式，简记作 $R \in$ 3NF。

【例 13-17】 关系 ST(学号，楼号，收费)；其中包含的最小函数依赖集为：

学号→楼号，楼号→收费。

函数依赖“学号→收费”也成立，但因为“收费”不是直接而是传递函数依赖于“学号”，所以这是一个冗余的函数依赖。

对上述关系模式进行分解，分解为两个关系：ST1(学号，楼号)和 ST2(楼号，收费)。

推论：如果关系模式 $R\in 1NF$，且它的每一个非主属性既不部分依赖也不传递函数依赖于码，则 $R\in 3NF$。

通过 3NF 的定义，可以得出这样的推论：不存在非主属性的关系模式一定属于 3NF。此推论由读者自行证明。

关系的规范化减少了冗余数据，节省了空间，避免了不合理的插入、删除、修改等操作，保持了数据的一致性；但是也导致了一些缺点，例如，信息放在不同表中，查询数据时有时需要把多个表连接在一起，增加了操作的时间和难度。因此关系模式要从实际设计的目标出发进行设计。

13.4.3 设计用户视图

视图是用户可直接访问的数据模式。在同一系统中，不同用户可有不同的视图。视图来自数据库基本表，但在结构和形式上可以不同于基本表，所以它不是基本表的简单的子集。

视图的作用主要包括：通过视图对基本表的屏蔽，为应用程序提供一定的逻辑独立性；可以更好地适应不同用户对数据的需求；为用户划定了访问数据的范围，有利于数据的保密等。

定义数据库基本表主要是从系统的时间效率、空间效率、易维护等角度出发。由于用户视图与基本表是相对独立的，因此在定义用户视图时可以注重考虑用户的习惯与方便，具体包括：

(1) 使用符合用户习惯的别名。

(2) 可以对不同级别的用户定义不同的视图，以保证系统的安全性。

(3) 简化用户对系统的使用。

如果某些局部应用中经常要使用某些很复杂的查询，为了方便用户，可以将这些复杂查询定义为视图，用户每次只对定义好的视图进行查询，从而简化用户的使用。

13.5 物理结构设计

数据库最终要存储在物理设备上。将逻辑设计中产生的数据库逻辑模型结合指定的DBMS，设计出最适合应用环境的物理结构的过程，称为数据库的物理结构设计。

数据库的物理结构设计分为两个步骤：

(1) 确定数据库的物理结构。

(2) 对所设计的物理结构进行评价。

如果所设计的物理结构的评价结果满足原设计要求，则可进入到物理实施阶段；否则，就需要重新设计或修改物理结构，有时甚至要返回逻辑设计阶段修改数据模型。

13.5.1 确定数据库的物理结构

数据库物理设计内容包括确定数据的存储结构、设计数据的存取路径、确定数据的存放

位置和确定系统配置。

1. 确定数据的存储结构

确定数据库存储结构时要综合考虑存取时间、存储空间利用率和维护代价3方面的因素。这3个方面常常是相互矛盾的,例如,消除一切冗余数据虽然能够节省存储空间,但往往会导致检索代价的增加,因此必须进行权衡,选择一个折中方案。确定数据的存储结构包括为各行记录分配连续或不连续的物理块等。

2. 设计数据的存取路径

DBMS常用存取方法有:B+树索引方法、聚簇(CLUSTER)方法和HASH(哈希)索引方法。

1) B+树索引方法

在关系数据库中,选择存取路径主要是指确定如何建立索引。例如,建立单列索引还是复合索引,应把哪些列作为主关键字、哪些列作为次关键字建立索引,建立多少个索引合适,是否建立聚集索引等等。

2) 聚簇方法

为了提高某个属性(或属性组)的查询速度,把这个或这些属性(称为聚簇码)上具有相同值的元组集中存放在连续的物理块,这称为聚簇。

聚簇的用途:大大提高按聚簇属性进行查询的效率;聚簇功能不但适用于单个关系,也适用于多个关系,假设用户经常要按系别查询学生成绩单,这一查询涉及学生关系和选修关系的连接操作,即需要按学号连接这两个关系,为提高连接操作的效率,可以把具有相同学号值的学生元组和选修元组在物理上聚簇在一起。这就相当于把多个关系按"预连接"的形式存放,从而大大提高了连接操作的效率。

3) HASH索引方法

有些数据库管理系统提供了HASH索引方法。由于HASH索引比较的是进行HASH运算之后的HASH值,所以,满足下列两个条件之一,此关系才可以选择HASH索引方法:

① 该关系的属性主要出现在等值连接条件中或主要出现在相等、不等或属于比较选择条件中。HASH索引在这几种情况中的查询效率要远高于B+树索引。

② 该关系的大小可预知,且关系的大小不变或该关系的大小动态改变,但所选用的DBMS提供了动态HASH索引方法。

3. 确定数据的存放位置

为了提高系统性能,数据应该根据应用情况将易变部分与稳定部分分磁盘存放、经常存取部分和存取频率较低部分分磁盘存放以及数据表和索引分磁盘存放、数据和日志分磁盘存放等等。

4. 确定系统配置

DBMS产品一般都提供了一些存储分配参数,供设计人员和DBA对数据库进行物理优化。初始情况下,系统为这些变量都赋予了合理的默认值。但是这些值不一定适合每一种应用环境。

对系统配置的变量,例如,同时使用数据库的用户数、同时打开的数据库对象数、缓冲区分配参数、物理块装填因子、数据库的大小、锁的数目等,在物理设计时应根据应用环境确定

这些参数值，以使系统性能最佳。

13.5.2 评价物理结构

数据库物理设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡，其结果可以产生多种方案，数据库设计人员必须对这些方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。

评价物理数据库的方法完全依赖于所选用的DBMS，主要是从定量估算各种方案的存储空间、存取时间和维护代价入手，对估算结果进行权衡、比较，选择出一个较优的物理结构。如果该结构不符合用户需求，则需要修改设计。

13.6 数据库实施和运行、维护

在数据库正式投入运行之前，还需要完成很多工作。比如，在模式和子模式中加入数据库安全性、完整性的描述，完成应用程序和加载程序的设计，数据库系统的试运行，并在试运行中对系统进行评价。如果评价结果不能满足要求，还需要对数据库进行修正设计，直到满意为止。数据库正式投入使用，并不意味着数据库设计生命周期的结束，而是数据库维护阶段的开始。

13.6.1 数据库实施

根据逻辑和物理设计的结果，在计算机上建立起实际的数据库结构，并装入数据，进行试运行和评价的过程，叫作数据库的实施（或实现）。

1. 建立实际的数据库结构

用DBMS提供的数据定义语言，编写描述逻辑设计和物理设计结果的程序（一般称为数据库脚本程序），经计算机编译处理和执行后，就生成了实际的数据库结构。

2. 数据加载

数据库应用程序的设计应该与数据库设计同时进行。一般地，应用程序的设计应该包括数据库加载程序的设计。在数据加载前，必须对数据进行整理。由于用户缺乏计算机应用背景的知识，常常不了解数据的准确性对数据库系统正常运行的重要性，因而未对提供的数据做严格的检查。所以，数据加载前，要建立严格的数据登录、录入和校验规范，设计完善的数据校验与校正程序，排除不合格数据。

3. 数据库试运行和评价

当加载了部分必需的数据和应用程序后，就可以开始对数据库系统进行联合调试，称为数据库的试运行。一般将数据库的试运行和评价结合起来，目的是：

（1）测试应用程序的功能；

（2）测试数据库的运行效率是否达到设计目标，是否为用户所接受。

测试的目的是为了发现问题，而不是为了说明能实现哪些功能。所以，测试中一定要有非设计人员的参与。

对于数据库系统的评价比较困难。需要估算不同存取方法的CPU服务时间及I/O服务时间。为此，一般还是从实际试运行中进行估价，确认其功能和性能是否满足设计要求，

对空间占用率和时间响应是否满意等等。

13.6.2 数据库的运行与维护

数据库试运行结果符合设计目标后,数据库就可以真正投入运行了。数据库投入运行标志着开发任务的基本完成和维护工作的开始。

对数据库设计进行评价、调整、修改等维护工作是一个长期的任务,也是设计工作的继续和提高。

概括起来,维护工作包括以下内容:

数据库的转储和恢复;数据库的安全性和完整性控制;数据库性能的监督、分析和改造;数据库的重组织和重构造。

13.7 数据库设计实例

13.7.1 银行卡管理系统数据库设计

1. 需求分析

与用户协商,了解用户的需求,需要哪些数据和操作(主要是查询),确定系统中应包含以下实体:储户、账户、交易记录。

储户的属性确定为:身份证号、姓名、电话;账户的属性确定为:账号、开户日期、类型、币种、密码、余额;交易记录的属性确定为:交易时间、账号、支出、收入、对方账号、交易地点、摘要。其中,每个储户可以拥有多个账户;每个账户拥有多个交易记录。

2. 概念结构设计

画出银行卡管理系统的 E-R 图,如图 13-13 所示。

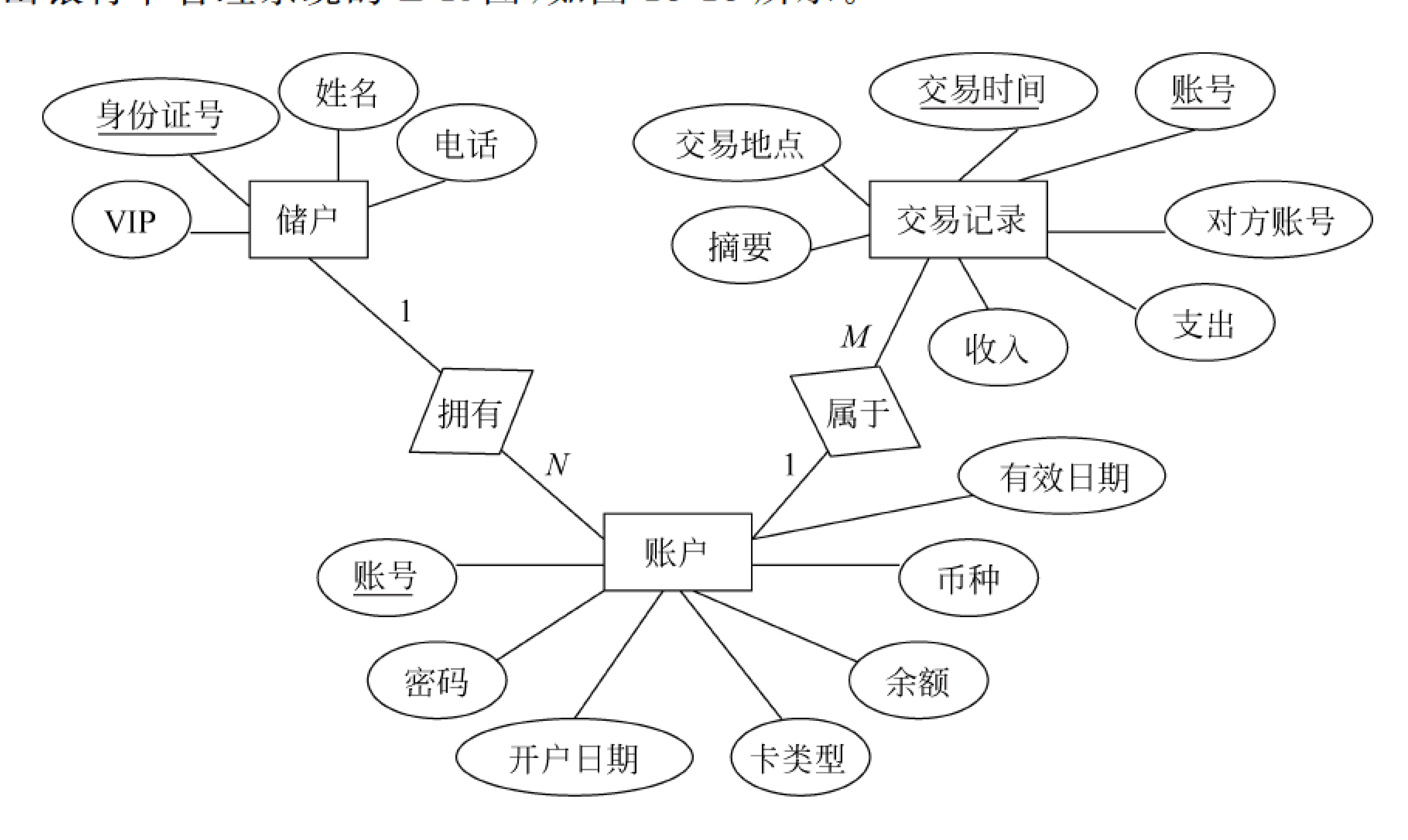


图 13-13 银行卡管理系统的 E-R 图

3. 逻辑结构设计

根据前面的转换原则,银行卡管理系统的关系模式设计如下:

储户(身份证号,姓名,电话)
账户(账号,身份证号,密码,开户日期,类型,币种,余额)
交易记录(交易时间,账号,支出,收入,对方账号,交易地点,摘要)

表结构设计,如表 13-5～表 13-7 所示。

表 13-5　储户表的表结构

属性名	类型	宽度	主码	取 值 范 围
身份证号	字符型	18	是	
姓名	字符型	10		
电话	字符型	11		数字字符
VIP	字符型	1		“是”或“否”

表 13-6　账户表的表结构

属性名	类型	宽度	主码	取 值 范 围
账号	字符型	20	是	数字字符
身份证号	字符型	18		参考储户表主码
密码	字符型	6		数字字符
开户日期	日期型			默认值：当前日期
类型	字符型	3		信用卡、借记卡等
币种	字符型	3		人民币、美元等
余额	货币型			

表 13-7　交易记录表的表结构

属性名	类型	宽度	主码	取 值 范 围
账号	字符型	20	是	数字字符,参考账户表主码
交易时间	日期型		是	
支出	货币型			
收入	货币型			
对方账号	字符型	3		数字字符
交易地点	字符型	30		
摘要	字符型	20		转账、消费、工资、劳务等

4. 物理结构设计

根据查询需求设计每一个关系的 B+树索引文件。在每个表上都按主码建立聚集索引；在账户表上分别按身份证号和开户日期建立非聚集索引；在交易记录表上分别按账号和交易时间建立非聚集索引。

13.7.2　钢材仓库管理系统设计

1. 需求分析

与用户协商,了解用户的需求,需要哪些数据和操作(主要是查询),确定系统中应包含以下实体：职工、仓库、钢材和供应商。

职工的属性确定为：工号、姓名、性别、出生年月、工种(销售员、采购员、仓库管理员)；

仓库的属性确定为：仓库编号、仓库名称、地址、联系电话、容量；钢材的属性确定为：钢材号、钢材名、品种、规格；供应商的属性确定为：供应商编号、供应商名称、地址、电话、联系人。

其中，一种钢材可以存放于多个仓库内，一个仓库也可以存放多种钢材；一个供应商可以供应多种钢材，一种钢材也可以由多个供应商提供，每个供应商供应一种钢材有一个报价；钢材、仓库与销售员之间有销售关系，它们是多对多关系，每个销售员销售每个仓库的每种钢材都有一个销售单号、出库数量和出库日期；采购员、钢材与仓库之间有采购关系，它们是多对多关系，每个采购员采购每种钢材都有一个入库单号、入库数量和入库日期；每个仓库有多名管理员，每个管理员只能管理一个仓库。

2. 概念结构设计

画出钢材仓库管理系统的 E-R 图，如图 13-14 所示。

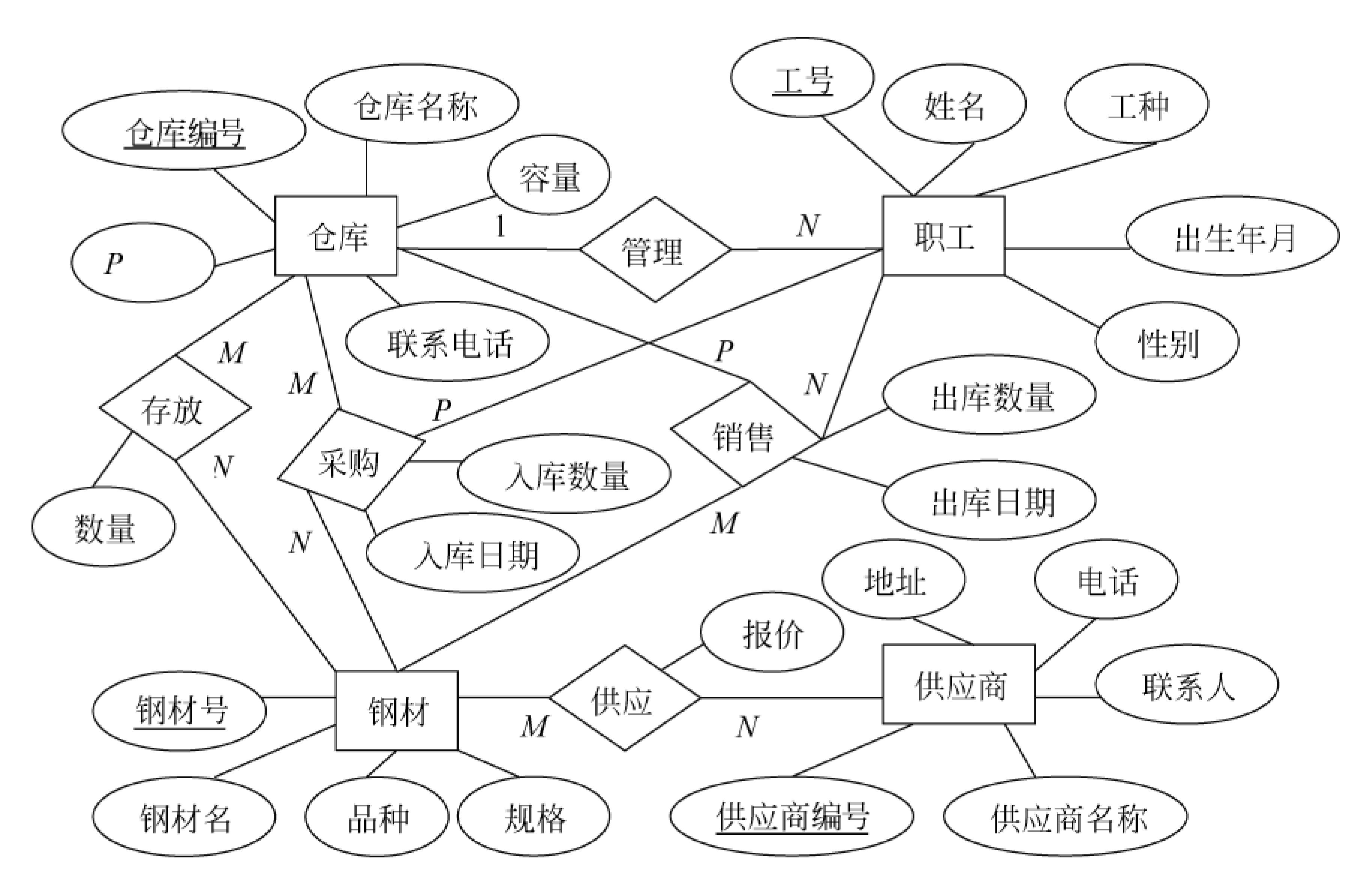


图 13-14 钢材仓库管理系统的 E-R 图

3. 逻辑结构设计

根据前面的转换原则，钢材仓库管理系统的关系模式设计，如下：

职工(工号，姓名，性别，出生年月，工种，仓库编号)
仓库(仓库编号，仓库名称，地址，联系电话，容量)
钢材(钢材号，钢材名，品种，规格)
供应商(供应商编号，供应商名称，地址，电话，联系人)
存放(仓库编号，钢材号，存放数量)
供应(供应商编号，钢材号，报价)
销售(出库单号，钢材号，仓库编号，销售员工号，出库数量，出库日期)
采购(入库单号，钢材号，仓库编号，采购员工号，入库数量，入库日期)

其中，职工关系中如果职工的工种为仓库管理员，则仓库编号属性的取值为仓库关系中仓库编号属性的某个值，否则为空值。

表结构设计如表 13-8～表 13-15 所示。

表 13-8　职工表的表结构

属性名	类型	宽度	主码	取 值 范 围
工号	字符型	5	是	
姓名	字符型	10		
性别	字符型	2		(男,女)
出生年月	日期型	20		1956-1-1～2050-1-1
工种	字符型	10		(销售员,采购员,仓库管理员)
仓库编号	字符型	4		参考仓库表主码

表 13-9　仓库表的表结构

属性名	类型	宽度	主码	取 值 范 围
仓库编号	字符型	4	是	
仓库名称	字符型	20		
地址	字符型	20		
联系电话	字符型	10		数字字符
容量	整型			＞＝总存放数量

表 13-10　钢材表的表结构

属性名	类型	宽度	主码
钢材号	字符型	4	是
钢材名	字符型	20	
品种	字符型	10	
规格	字符型	10	

表 13-11　供应商表的表结构

属性名	类型	宽度	主码	取 值 范 围
供应商编号	字符型	5	是	
供应商名称	字符型	30		
地址	字符型	40		
电话	字符型	11		数字字符
联系人	字符型	10		

表 13-12　存放表的表结构

属性名	类型	宽度	主码	取 值 范 围
仓库编号	字符型	4	是	参考仓库表主码
钢材号	字符型	4	是	参考钢材表主码
存放数量	整型			

表 13-13　供应表的表结构

属性名	类型	宽度	主码	取 值 范 围
供应商编号	字符型	5	是	参考供应商表主码
钢材号	字符型	4	是	参考钢材表主码
报价	实型			0.00～9999.99

表 13-14　销售表的表结构

属性名	类型	宽度	主码	取 值 范 围
出库单号	字符型	10	是	
仓库编号	字符型	4		参考仓库表主码
销售员工号	字符型	5		参考员工表主码
钢材号	字符型	4		参考钢材表主码
出库数量	整型			<=存放数量
出库日期	日期型			2000-1-1～2030-1-1

表 13-15　采购表的表结构

属性名	类型	宽度	主码	取 值 范 围
入库单号	字符型	10	是	
仓库编号	字符型	4		参考仓库表主码
采购员工号	字符型	5		参考员工表主码
钢材号	字符型	4		参考钢材表主码
入库数量	整型	2		
入库日期	日期型	8		2000-1-1～2030-1-1

4. 物理结构设计

根据查询需求设计每一个关系的B+树索引文件。在每个表上都按主码建立聚集索引；在职工表上按姓名建立非聚集索引；在仓库表上按仓库名称建立非聚集索引；在钢材表上分别按钢材名、品种和规格建立非聚集索引；在供应商表上按供应商名称建立非聚集索引；在供应表上按钢材号和报价的组合建立非聚集复合索引；在销售表上分别按仓库编号、销售员工号、钢材号和出库日期建立非聚集索引；在销售表上分别按仓库编号、采购员工号、钢材号和入库日期建立非聚集索引。

习　题　13

1. 简述数据库设计的概念。
2. 简述数据库设计过程。
3. 简述数据库设计过程各个阶段的设计任务。
4. 简述需求分析的重要性。
5. 名词解释：实体、属性、码、联系。
6. 简述E-R图的通用表示形式。
7. 简述数据库概念结构设计的重要性和设计步骤。
8. 什么是数据库的逻辑结构设计？试述其设计步骤。
9. 简述把E-R图转换为关系模型的转换规则。
10. 名词解释：函数依赖、非平凡函数依赖、平凡的函数依赖、部分函数依赖、完全函数依赖、传递函数依赖、范式、第一范式、第二范式、第三范式。
11. 非规范的关系模式存在哪些问题？
12. 简述视图的作用。

13. 将如图 13-15 所示学生信息数据库系统的 E-R 图转换为关系模型。

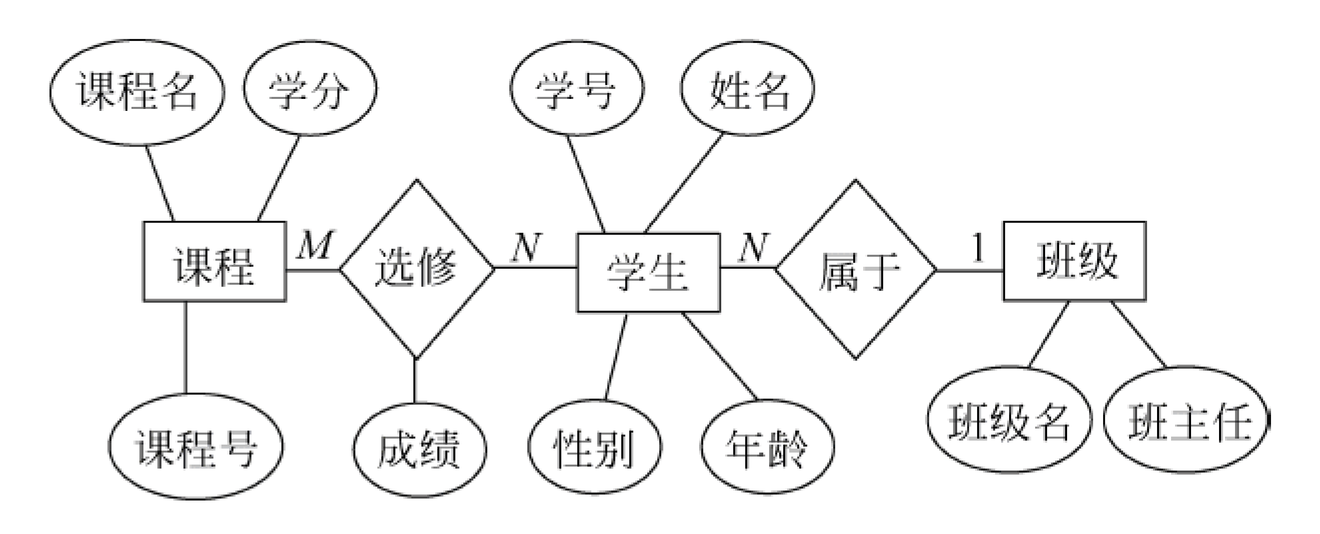


图 13-15　习题 13 图

14. 一个设备销售管理系统中有如下信息：

设备实体包含设备编号、设备名称、型号规格、数量属性；部门实体包含部门编号、部门名称、部门经理、电话属性；客户实体包含客户编号、客户名称、地址、电话属性。

其中，任何设备都可以销售给多个客户，每个客户购买一种设备都要登记购买数量；一个部门可以管理多种设备，一种设备仅由一个部门来调动管理。

根据以上情况，完成如下设计：

(1) 设计系统的 E-R 图；

(2) 将 E-R 图转换为关系模式，标出每个关系模式的主码；

(3) 判断所有的关系模式是否都达到了 3NF，对没有达到的进行规范化。

15. 某电子商务网站要求提供下述服务：

可随时查询库存中现有物品的名称、数量和单价，所有物品均应由物品编号唯一标识；可随时查询顾客订货情况，包括顾客号、顾客名、所订物品编号、订购数量、联系方式、交货地点，所有顾客编号不重复；当需要时，可通过数据库中保存的供应商名称、电话、邮编与地址信息向相应供应商订货，一个编号货物只由一个供应商供货。

根据以上要求，完成如下任务：

(1) 根据语义设计出 E-R 模型；

(2) 将该 E-R 模型转换为一组等价的关系模式，并标出各关系模式的主码；

(3) 判断所有的关系模式是否都达到了 3NF，对没有达到的进行规范化。

第 14 章 基于 C♯.NET 的数据库应用系统开发

SQL Server 2008 作为一个数据库管理系统，最终要向应用程序提供数据，供用户使用。所以数据库的开发是数据库系统必不可少的内容。

ASP.NET 是一种用于创建基于 Web 的应用程序的编程模型。它在 Web 服务器的环境中运行，例如 Microsoft Internet Information Server (Microsoft 互联网信息服务器，IIS)，并且根据浏览器请求指示在服务器上执行程序。

C♯语言是 ASP.NET 平台的第一语言，也是目前程序开发人员使用最广泛的开发工具之一。因此，如何使用 C♯语言开发数据库应用程序是软件开发人员最有必要了解的技术之一。

本章首先介绍 C♯语言及 ASP.NET 中访问数据库的组件 ADO.NET，然后介绍一个基于 C♯.NET 的数据库系统开发实例“教学管理系统”。

14.1 C♯语言简介

20 世纪 80 年代以来，C/C++语言一直是使用最广泛的商业化开发语言。但在带来强大控制能力和高度灵敏性的同时，其代价是相对较长的学习周期和较低的开发效率，同时对控制能力的滥用也给程序的安全性带来了潜在的威胁。C++语言过度的功能扩张也破坏了面向对象的设计理念。因此，软件行业迫切需要一种全新的现代程序设计语言，它能够在控制能力与生产效率之间达到良好的平衡，特别是将高端应用开发与底层平台访问紧密结合在一起，并与 Web 标准保持同步。C♯(读作 C-Sharp)语言就是这一使命的承担者。

C♯语言是从 C/C++语言发展而来的，它汲取了包括 C++、Java、Delphi 在内的多种语言的精华，是一种简单易学、类型安全和完全面向对象的高级程序设计语言。它的设计目标就是在继承 C/C++强大功能的用时，兼有 RAD(快速应用程序开发)语言的高效性。作为.NET 的核心编程语言，C♯充分享受了 CLR 所提供的优势，能够与其他应用程序方便地集成和交互。

C♯语言的突出特点包括：

(1) 语法简洁。C♯取消了指针，也不定义烦琐的伪关键字；它使用有限的指令、修饰符和操作符，语法上几乎不存在任何冗余，整个语言结构十分清晰。初学者通常能够快速掌握 C♯的基本特性，而 C/C++程序员转入 C♯则几乎不会有什么障碍。

(2) 完全面向对象。C♯具有面向对象的语言所应有的基本特性：封装、继承和多态性。它禁止多继承，禁止各种全局方法、全局变量和常量。C♯以类为基础来构建所有的类型，并通过命名空间对代码进行层次化的组织和管理。许多精巧的对象设计模式都在 C♯

语言中得到了有效的应用。

(3) 与 Web 紧密结合。借助 Web 服务框架,C#使得网络开发和本地开发几乎一样简单。开发人员无须了解网络的细节,可以用统一的方式来处理本地的和远程的 C#对象,而 C#组件能够方便地转为 Web 服务,并被其他平台上的各种编程语言调用。

(4) 目标软件的安全性。C#符合通用类型系统的类型安全性要求,并用 CLR 所提供的代码访问安全特性,从而能够在程序中方便地配置安全等级和用户权限。此外,垃圾收集机制自动管理对象的生命周期,这使得开发人员无须再负担内存管理的任务,应用程序的可靠性进一步得到提高。

(5) 版本管理技术。C#语言中内置了版本控制功能,并通过接口和继承来实现应用的可扩展性。应用程序的维护和升级更加易于管理。

(6) 灵活性与兼容性。C#允许使用非托管代码来与其他程序(包括 COM 组件、WIN32API 等)进行集成和交互。它还可以通过委托(delegate)来模拟指针的功能,通过接口来模拟多继承的实现。

14.2 使用 ADO.NET 访问 SQL Server 数据库

ADO.NET 是.NET Framework 中的一套类库,它可让用户更加方便地在应用程序中使用数据。Microsoft 收集了过去几十年中最佳的数据连接的操作实践,并编写代码实现这些实践。这些代码被包装进了一些对象中,以便其他软件可以方便地使用。

ADO.NET 中的代码处理了大量的数据库特有的复杂情况,所以当 ASP.NET 页面设计人员想读取或者写入数据时,他们只需编写少量的代码,并且这些代码都是标准化的。就像 ASP.NET 一样,ADO.NET 不是一种语言,它是对象(类)的集合,在对象(类)中包含了由 Microsoft 编写的代码。可以使用诸如 C#或者 Visual Basic 等编程语言来在对象外部运行这些代码。

可以将 ADO.NET 看作是一个介于数据源和数据使用者之间的非常灵巧的转换层。ADO.NET 可以接收数据使用者语言中的命令,然后将这些命令转换成在数据源中可以正确执行任务的命令。另外,ASP.NET 4.5 提供了服务器端数据控件,可以更方便地与 ADO.NET 交互工作,这减少了直接使用 ADO.NET 对象的需求。

14.2.1 ADO.NET 的对象模型

ADO.NET 是.NET 应用程序的数据访问模型,它能用于访问关系型数据库系统。ADO.NET 对象模型有 5 个主要的组件,分别是 Connection 对象、Command 对象、DataReader 对象、DataSet 对象以及 DataAdapter 对象。

上面的几个对象组成的数据操作组件最主要的是当作 DataSet 对象以及数据源之间的桥梁,负责将数据源中的数据读取到 DataSet 对象中,以及将数据存回到数据源中的工作。ADO.NET 的对象结构模型示意图如图 14-1 所示。

1. Connection(连接)对象

Connection 对象表示与数据源之间的连接,用它来建立或断开与数据库的连接。Connection 对象起到渠道的作用,其他对象(如 DataAdapter 和 Command 对象)通过它与

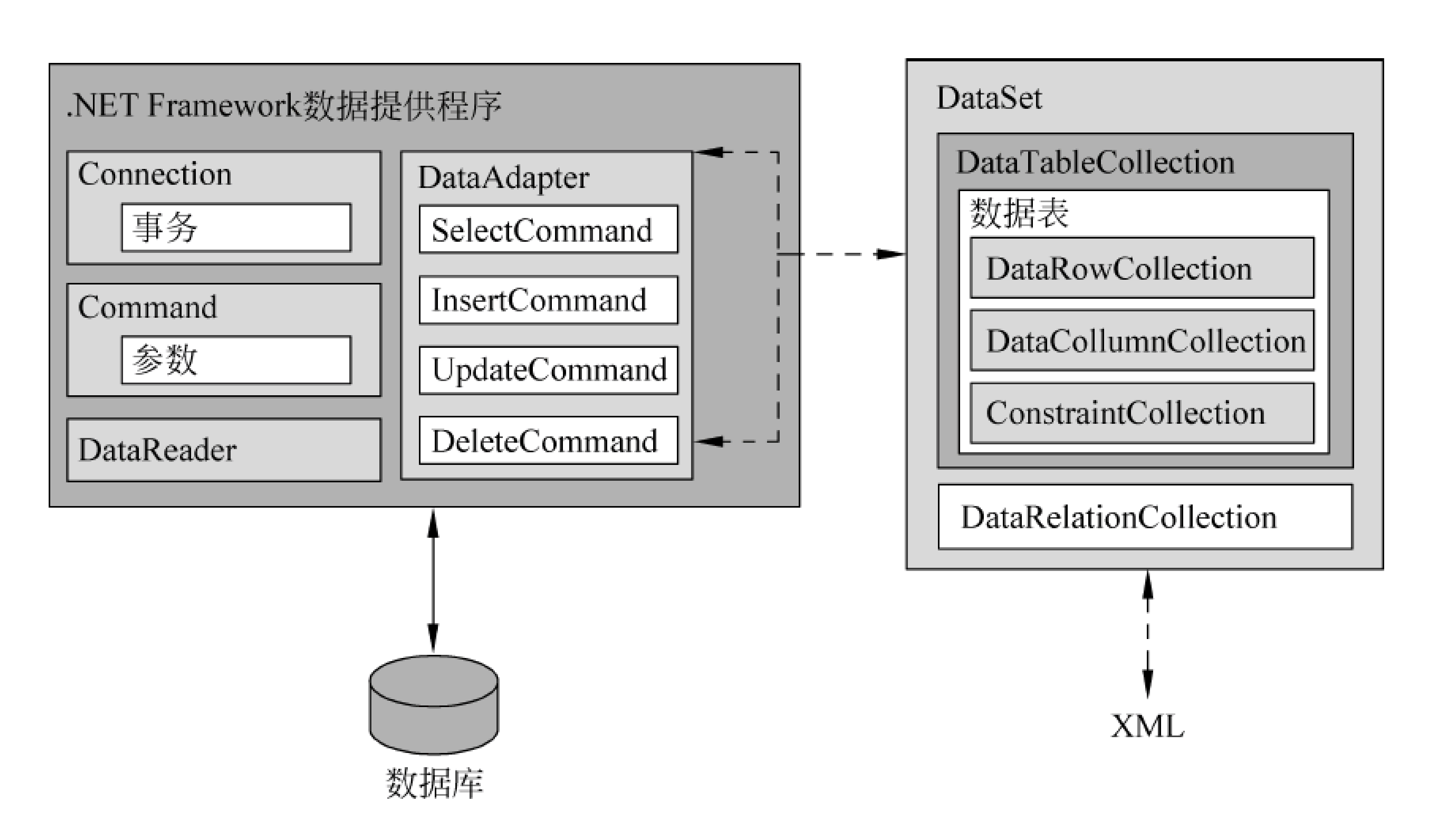


图 14-1　ADO. NET 的对象结构模型示意图

数据库通信，以提交查询并获取查询结果。

Connection 对象提供了对数据源连接的封装，其中包括连接方法及描述当前连接状态的属性。在 Connection 类中最重要的属性是 ConnectionString（连接字符串），该属性用来指定服务器名称、数据源信息及其他登录信息。

ConnectionString 中有两个重要的部分：字符串的内容和数据提供器名称。字符串内容中包含有 DataSource（数据源）、Initial Catalog（默认连接数据库）及用于描述用户身份的 User ID 和 Password。

2. Command（命令）对象

Command 对象主要可以用来对数据库发出一些指令，例如可以对数据库下达查询、插入、修改、删除等数据指令，以及呼叫存在数据库中的预存程序等。这个对象架构在 Connection 对象上，也就是 Command 对象是通过连接到数据源的 Connection 对象来下命令的。所以 Connection 连接到哪个数据库，Command 对象的命令就下到哪里。

数据库支持多种不同类型的查询。有些查询通过引用一个或多个表、视图或者是通过调用一个存储过程来获取数据行，有些查询会对数据行进行修改，还有一些查询通过创建或修改诸如表、视图或存储过程等对象对数据库的结构进行有关操作。可使用 Command 对象对数据库执行任何一种查询操作。

使用 Command 对象查询数据库相当简单。先将 Connection 属性设置为连接数据库的 Connection 对象，然后在 CommandText 属性中指定查询文本即可。

3. DataReader 对象

DataReader 用于以最快的速度检索并检查查询所返回的行。可使用 DataReader 对象来检查查询结果，一次检查一行。当移向下一行时，前一行的内容就会被放弃。由 DataReader 返回的数据是只读的，不支持更新操作。所以 DataReader 对象使用起来不但节省了资源而且效率高。另外，因为 DataReader 对象不用把数据全部传回，所以降低了网络的负载。

4. DataSet 对象

DataSet 对象可视为暂存区，可以把数据库中查到的信息保存起来，甚至可以显示整个

数据库。从其名称可以看出，DataSet 对象包含一个数据集。可以将 DataSet 对象视为许多 DataTable 对象(存储在 DataSet 对象的 Tables 集合中)的容器。

创建 ADO. NET 的目的是帮助开发人员建立大型的多层数据库应用程序。有时，开发人员可能希望访问一个运行在中间层服务器上的组件，以获取许多表的内容。这时不必重复调用该服务器以便每次从一个表中获取数据，而是可以将所有数据都封装入一个 DataSet 对象之中，并在一次单独调用中将其返回。但 DataSet 对象的功能绝不仅仅是作为多个 DataTable 对象的容器。

存储在 DataSet 对象中的数据未与数据库连接。对数据所做的任何更改都将只是缓存在每个 DataRow 之中。要将这些更改传递给数据库时，将整个 DataSet 回传给中间层服务器可能并非一种有效方法。可以使用 GetChanges 方法仅从 DataSet 中选出被修改的行。通过这样的方式，可以在不同进程或服务器之间传递较少数据。

5. DataAdapter 对象

DataAdapter 对象充当数据库和 ADO. NET 对象模型中非连接对象之间的桥梁。DataAdapter 对象类的 Fill 方法提供了一种高效机制，用于将查询结果引入 DataSet 或 DataTable 中，以便能够脱机处理数据。还可以利用 DataAdapter 对象向数据库提交存储在 DataSet 对象中的挂起更改。

利用 DataAdapter 对象，可以设置 UpdateCommand、InsertCommand 以及 DeleteCommand 属性来调用存储过程，这些存储过程将修改、添加或删除数据库中相应表的数据行。然后可以只调用 DataAdapter 对象的 Update 方法，ADO. NET 就会使用所创建的 Command 对象向数据库提交 DataSet 中缓存的更改。

14.2.2 利用 ADO. NET 访问数据库的基本操作

ADO. NET 提供了两种访问数据库的方法，如图 14-2 所示。

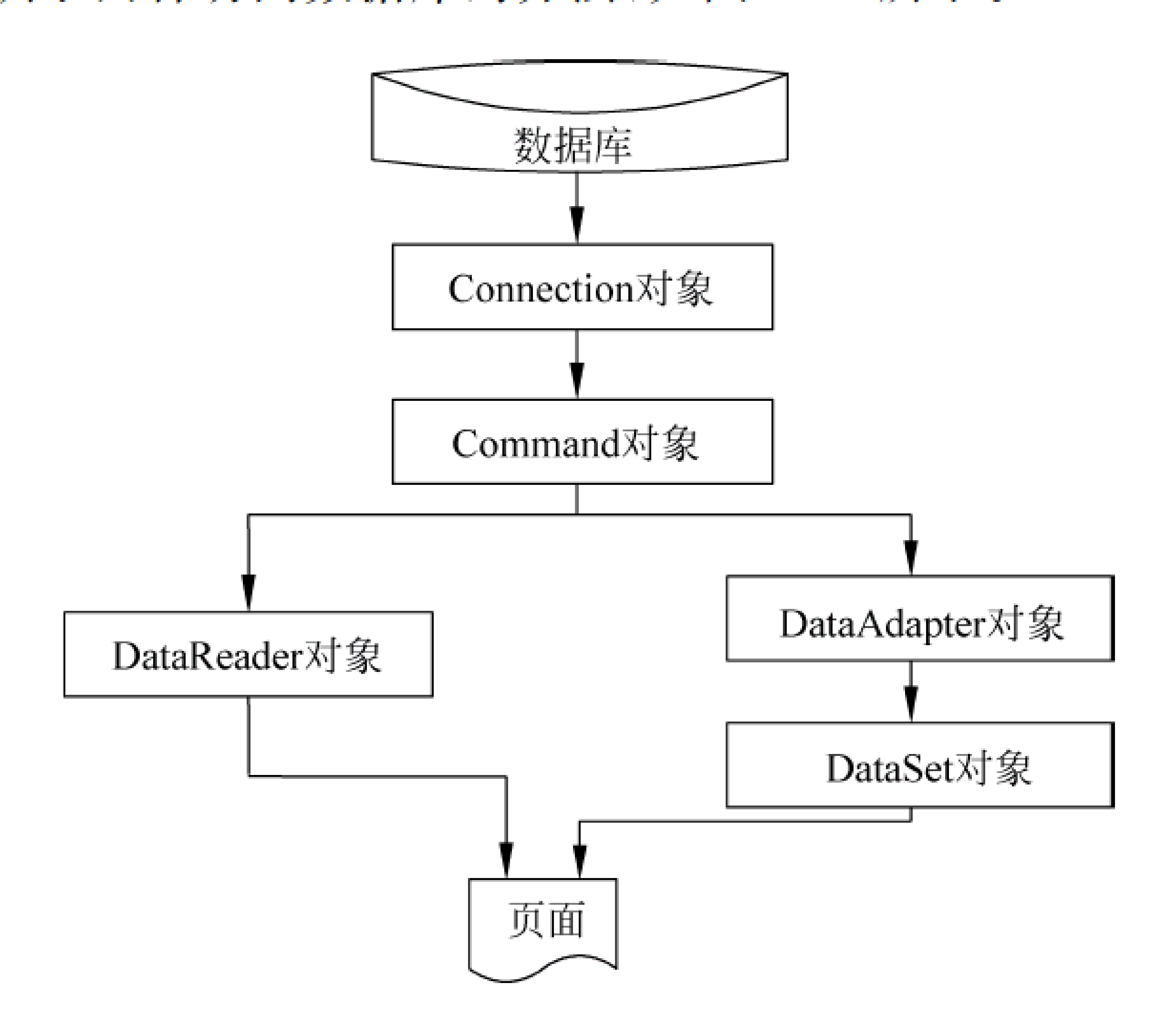


图 14-2 ADO. NET 访问数据库的方法

利用 Connection、Command 和 DataReader 对象访问数据库，只能从数据库读取数据，不能添加、修改和删除记录。如果只想进行查询，这种方式效率更高一些。

利用 Connection、Command、DataAdapter 和 DataSet 对象访问数据库，这种方式比较灵活，不仅可以对数据库进行查询操作，还可以进行增加、删除和修改等操作。

访问数据库中的数据，首先要实现数据库的连接，然后才能实现对数据库的各种操作。下面以访问 SQL Server 数据库为例，介绍 ADO. NET 访问数据库的基本步骤。

① 引入数据命名空间和 SQL 命名空间，语句分别为：using System. Data 和 using System. Data. SqlClient。

② 创建连接对象并实例化，填充连接字符串变量以存放建立连接所需要的信息。例如，连接一个名为 teaching 的 SQL Server 数据库的语句：

```
SqlConnection con = new SqlConnection( );
con.ConnectionString = @"Data Source = .\SQLEXPRESS; AttachDbFilename = E:\data\ teaching_
Data.MDF; Integrated Security = True;Connect Timeout = 30;User Instance = True";
```

③ 打开数据库连接。

```
con.Open( );
```

④ 使用连接。从数据源中读取数据或向数据源中写入数据。具体实现依据执行的 SQL 操作不同而有所区别。

首先要创建命令对象并实例化，填充命令字符串变量以存放对数据库的操作命令，即 SQL 语句。

如果只是从数据库读取数据，可以使用 DataReader 对象和 Command 对象的 ExecuteReader( )方法访问数据库；如果对数据库进行增加、删除或修改等操作，可以直接使用 Command 对象的 ExecuteNonQuery( )方法；如果想在内存中直接操作数据库的数据，可以使用 DataAdapter 和 DataSet 对象，实现对数据库的查询和进行增加、删除或修改等操作。

例如，只是从数据库读取数据，其语句序列如下：

```
SqlCommand cmd = con.CreateCommand();         //创建命令对象并实例化
cmd.CommandText = "select * from student"; //SqlCommand 的属性 CommandText 是一条 SQL 语句,
                                              //student 为所连接数据库中的表
SqlDataReader dr = cmd.ExecuteReader();       //建立 DataReader 对象迅速获取查询结果
```

也可以利用.NET 中的控件来实现数据库中数据的操作，比如 GridView、FormView、DetailsView 和 DataList 控件等等。主要数据控件如图 14-3 所示。

以 GridView 控件为例，显示数据库表中数据的步骤如下：

① 在配置文件(web. config)中，加入下面一段语句：

```
<connectionStrings>
  <add name = "TeachingConnectionString" connectionString = "Data Source = (LocalDB)\v11.0;
AttachDbFilename = |DataDirectory|\Teaching.mdf;Integrated Security = True; "
    providerName = "System.Data.SqlClient" />
</connectionStrings>
```

② 为 GridView 新建数据源：选择 TeachingConnectionString 连接串，设置 SQL 查询语句，比如 select * FROM student。

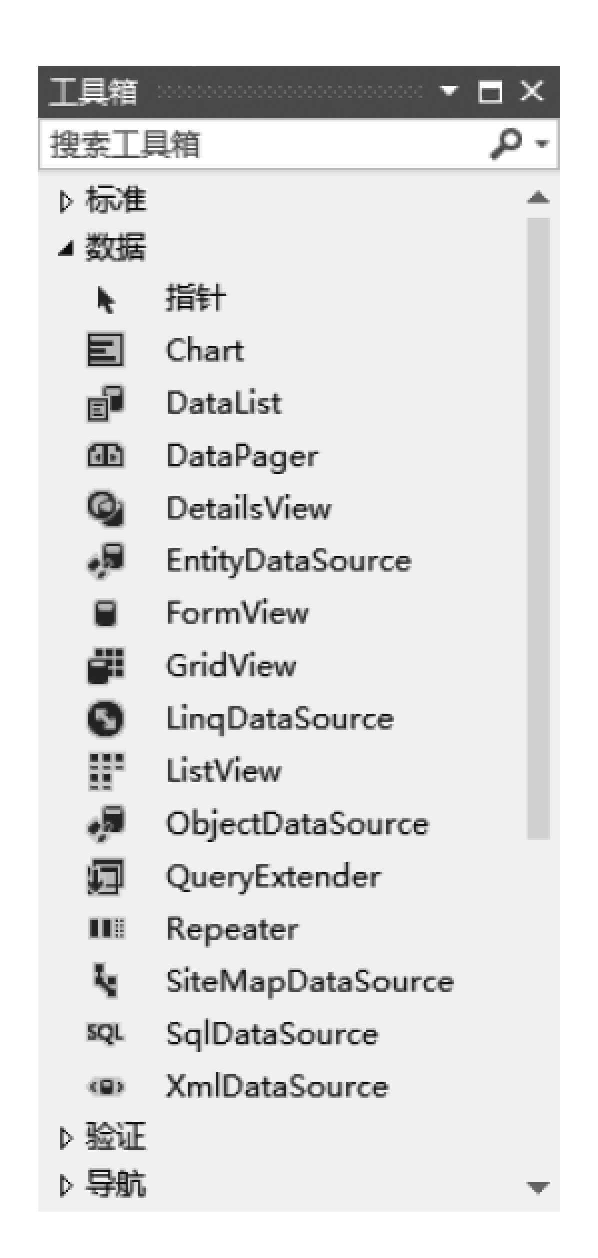


图 14-3 .NET 中的数据控件

③ 还可以对 GridView 控件进行属性设置，比如分页显示、排序显示等等。GridView 控件显示数据库表中数据的样式如图 14-4 所示。



图 14-4 GridView 控件显示数据库表中数据

14.3 数据库应用系统开发实例

本节主要通过一个完整的开发实例“教学管理系统”，讨论后台数据库使用 SQL Server、前台开发工具使用 ASP. NET(C＃语言)进行数据库系统开发的过程和方法。

14.3.1 数据库设计

数据库应用系统的开发也是一项软件工程，称为数据库工程。按照规范化设置的方法，

考虑数据库及其应用系统开发全过程，将“教学管理系统”数据库设计分为以下 6 个阶段：需求分析阶段、概念结构设计阶段、逻辑结构设计阶段、物理结构设计阶段、数据库实施阶段和运行维护阶段。

1. 需求分析

通过对教学管理的日常工作进行详细的调查和分析，确定“教学管理系统”实现的功能。通过该系统，学校不同角色的用户可以通过网络完成教学管理功能：管理员通过该系统实现对学生、教师、课程的添加和维护及学生成绩的维护，并根据需要进行某些信息的查询；教师通过该系统实现查看个人任课情况、课程选修情况和对课程成绩的录入；学生通过该系统实现选课和个人成绩查询等功能；还有公共模块，实现修改个人密码功能。

2. 概念结构设计

通过需求分析阶段的分析结果，本系统所要设计的实体和属性如下：

学生(学号,密码,姓名,性别,出生日期,专业,年级)
教师(教师号,密码,姓名,性别,学院)
课程(课程号,课程名,学分,学时)
管理员(管理员账号,姓名,密码)

系统的 E-R 图如图 14-5 所示，省略实体中的属性。

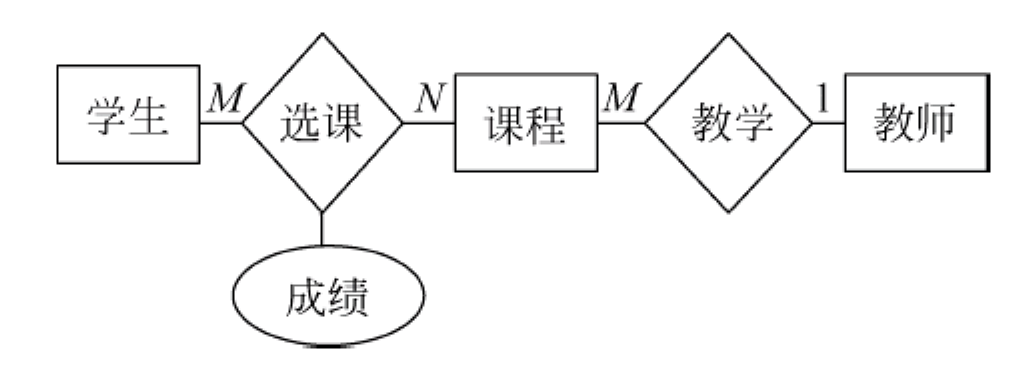


图 14-5　教学管理系统 E-R 图

3. 逻辑结构设计

将概念设计阶段的 E-R 图转换成关系模式，设计出“教学管理系统”的逻辑结构，并根据程序需要设计视图。

1）关系设计

根据转换原则转换成 5 个关系模式，关系的主码用下画线标出。

学生(学号,密码,姓名,性别,出生日期,年级,专业)
教师(教师号,密码,姓名,性别,学院)
课程(课程号,课程名,学分,学时,教师号)
选课(学号,课程号,成绩)
管理员(管理员账号,姓名,密码)

将关系模式转换为具体的 RDBMS 中支持的关系数据模型(表结构)。本系统在 SQL Server 数据库管理系统中共设计 5 个表，分别为学生表、教师表、课程表、选课表和管理员表，为保护数据的安全，我们将对学生、教师和管理员的密码进行加密保存，表结构见图 14-6～图 14-10。

列名	数据类型	允许 Null 值
学号	char(10)	☐
密码	varbinary(500)	☐
姓名	nvarchar(10)	☐
性别	nchar(1)	☐
出生日期	date	☐
专业	nvarchar(10)	☐
年级	nchar(5)	☐

图 14-6　学生表

列名	数据类型	允许 Null 值
课程号	char(16)	☐
课程名	nvarchar(10)	☐
学分	tinyint	☐
学时	tinyint	☐
教师号	char(5)	☐

图 14-7　课程表

列名	数据类型	允许 Null 值
学号	char(10)	☐
课程号	char(16)	☐
成绩	int	☑

图 14-8　选课表

列名	数据类型	允许 Null 值
教师号	char(5)	☐
密码	varbinary(500)	☐
姓名	nchar(10)	☐
性别	nchar(1)	☐
学院	nvarchar(10)	☐

图 14-9　教师表

列名	数据类型	允许 Null 值
管理员账号	char(5)	☐
姓名	nchar(10)	☐
密码	varbinary(500)	☐

图 14-10　管理员表

2）设计视图

视图 1(学生视图)：用于学生信息的查询和维护。创建代码如下：

```
CREATE VIEW student_view
AS SELECT     学号, 姓名, 性别, 出生日期, 专业, 年级, 密码
FROM          dbo.Student
```

视图 2(课程视图)：用于课程信息的查询和维护。创建代码如下：

```
CREATE VIEW course_view
AS SELECT     课程号, 课程名, 学分, 学时, 教师号
FROM          dbo.Course
```

视图 3(教师视图)：用于教师信息的查询和维护。创建代码如下：

```
CREATE VIEW teacher_view
AS SELECT     教师号, 密码, 姓名, 性别, 学院
FROM          dbo.Teacher
```

视图 4(选课视图)：用于选课信息的查询和维护。创建代码如下：

```
CREATE VIEW CS_view
AS SELECT     学号, 课程号, 成绩
FROM          dbo.CS
```

视图 5(选课人数视图)：用于管理员查询每门课的选修人数。创建代码如下：

```
CREATE VIEW [dbo].[Pnum]
AS SELECT Course.课程号,课程名,Course.教师号,Teacher.姓名,学院, count (CS.课程号) as 选课
人数
from CS right join Course on CS.课程号 = Course.课程号
left join Teacher on Teacher.教师号 = Course.教师号
group by Course.课程号,课程名,Course.教师号,Teacher.姓名,学院
```

视图 6(平均成绩视图)：用于管理员查询每门课的平均成绩。创建代码如下：

```
CREATE VIEW [dbo].[Avg]
```

```
AS SELECT Course.课程号,课程名,Course.教师号,Teacher.姓名,学院, avg (CS.成绩) as 平均成绩
from CS join Course on CS.课程号 = Course.课程号
join Teacher on Teacher.教师号 = Course.教师号
group by Course.课程号,课程名,Course.教师号,Teacher.姓名,学院
```

4. 物理结构设计

根据教学管理系统的数据操作需要,为各个表设计索引文件。每个表已经按主码自动创建了一个聚集索引,其他索引如下:

(1) 按学生表的"姓名"列升序创建一个非聚集索引。

(2) 按学生表的"年级"列升序、"专业"列升序创建一个非聚集索引。

(3) 按课程表的"课程名"列升序创建一个非聚集索引。

(4) 按课程表的"教师号"列升序创建一个非聚集索引。

(5) 按教师表的"姓名"列升序创建一个非聚集索引。

(6) 按教师表的"学院"列升序创建一个非聚集索引。

5. 数据库实施

在 SQL Server 2008 中创建 teaching 数据库,创建其中的 5 个表,并为表创建索引然后向表中添加数据,创建 6 个视图;再根据程序功能设计 15 个存储过程和 3 个表值函数。

(1) 存储过程 loginMan,用于实现管理员登录。

```
CREATE PROCEDURE [dbo].[loginMan]
@mno char(10),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
if convert(varchar(20),DECRYPTBYKEY(select 密码 from student where 管理员账号 = @mno)) =
@password
select 姓名 from manager where 管理员账号 = @mno
```

(2) 存储过程 loginStu,用于实现学生登录。

```
CREATE PROCEDURE [dbo].[loginStu]
@sno char(10),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
if convert(varchar(20),DECRYPTBYKEY(select 密码 from student where 学号 = @sno)) =
@password
select 姓名 from student where 学号 = @sno
```

(3) 存储过程 loginTea,用于实现教师登录。

```
CREATE PROCEDURE [dbo].[loginTea]
@tno char(10),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
if convert(varchar(20),DECRYPTBYKEY(select 密码 from student where 教师号 = @tno)) =
@password
select 姓名 from teacher where 教师号 = @tno
```

(4) 存储过程 addstudent,用于实现管理员添加学生信息。

```
CREATE PROCEDURE [dbo].[addstudent]
@sno char(10),@sname nvarchar(10),@ssex nchar(1),
@sbirthday date,@specialty nvarchar(10),@grade nchar(5),
@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
insert into student values (@sno,ENCRYPTBYKEY(KEY_GUID('PWDKEY'),
@password),@sname,@ssex,@sbirthday,@specialty,@grade)
```

(5) 存储过程 addteacher,用于实现管理员添加教师信息。

```
CREATE PROCEDURE [dbo].[addteacher]
@tno char(5),@tname nvarchar(10),@tsex nchar(1),
@institute nvarchar(10),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
insert into teacher values (@tno,ENCRYPTBYKEY(KEY_GUID('PWDKEY'),
@password),@tname,@tsex,@institute)
```

(6) 存储过程 maNo_Pwd,用于实现管理员修改密码时验证旧密码输入是否正确。

```
CREATE PROCEDURE [dbo].[maNo_Pwd]
@mno char(5),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
select 姓名 from manager where 管理员账号 = @mno and
convert(varchar(20),DECRYPTBYKEY(密码)) = @password
```

(7) 存储过程 stuNo_Pwd,用于实现学生修改密码时验证旧密码输入是否正确。

```
CREATE PROCEDURE [dbo].[stuNo_Pwd]
@sno char(10),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
select 姓名 from student where 学号 = @sno and
convert(varchar(20),DECRYPTBYKEY(密码)) = @password
```

(8) 存储过程 teaNo_Pwd,用于实现教师修改密码时验证旧密码输入是否正确。

```
CREATE PROCEDURE [dbo].[teaNo_Pwd]
@tno char(5),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
select 姓名 from teacher where 教师号 = @tno and
convert(varchar(20),DECRYPTBYKEY(密码)) = @password
```

(9) 存储过程 modiManPwd,用于实现管理员修改密码。

```
CREATE PROCEDURE [dbo].[modiManPwd]
@password varchar(20),@mno char(5)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
update ma_view set 密码 = ENCRYPTBYKEY(KEY_GUID('PWDKEY'),@password)
where 管理员账号 = @mno
```

(10) 存储过程 modiStuPwd,用于实现学生修改密码。

```
CREATE PROCEDURE [dbo].[modiStuPwd]
@password varchar(20),@sno char(10)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
update student_view set 密码 = ENCRYPTBYKEY(KEY_GUID('PWDKEY'),@password)
where 学号 = @sno
```

(11) 存储过程 modiTeaPwd,用于实现教师修改密码。

```
CREATE PROCEDURE [dbo].[modiTeaPwd]
@tno char(5),@password varchar(20)
as
OPEN SYMMETRIC KEY PWDKEY
DECRYPTION BY CERTIFICATE cert_Pwd
update teacher_view set 密码 = ENCRYPTBYKEY(KEY_GUID('PWDKEY'),@password)
where 教师号 = @tno
```

(12) 存储过程 teacherQElect,用于实现教师查询某门课学生选课情况。

```
CREATE PROCEDURE [dbo].[teacherQElect] @cno char(16)
AS if (select count( * )from Student,CS
      where Student.学号 = CS.学号 and 课程号 = @cno)> 0
  select student.学号,姓名,年级,专业 from Student,CS
  where Student.学号 = CS.学号 and 课程号 = @cno
  order by 专业,年级
else
  SELECT 0 as 选课人数
RETURN
```

(13) 存储过程 teacher_cNO_Name,用于实现教师选择要查询学生选课情况的课程。

```
CREATE PROCEDURE [dbo].[teacher_cNO_Name] @tno char(5)
AS
select distinct Course.课程号,课程名 from Course,CS where Course.课程号 = CS.课程号
and 教师号 = @tno
```

(14) 存储过程 teacher_c_score,用于实现教师提交成绩时显示某门课的学生基本信息和成绩。

```
CREATE PROCEDURE [dbo].[teacher_c_score] @cno char(16)
```

```
AS
SELECT Student.学号,姓名,专业,年级,成绩 FROM Student,CS
WHERE Student.学号 = CS.学号 AND 课程号 = @cno
```

(15) 存储过程 teacher_cs_NO_Name,用于实现教师选择要提交成绩的课程。

```
CREATE PROCEDURE [dbo].[teacher_cs_NO_Name] @tno char(5)
AS
select distinct Course.课程号,课程名 from Course,CS where Course.课程号 = CS.课程号
and 教师号 = @tno
```

(16) 表值函数 stuscore,用于实现学生查询考试成绩。

```
CREATE FUNCTION [dbo].[stuscore](@sno varchar(11))
RETURNS table
AS
RETURN
  select CS.课程号,课程名,成绩 from CS,Course
  where Course.课程号 = CS.课程号 and 学号 = @sno
```

(17) 表值函数 SeltCourse1,用于实现学生选课时显示该学生还未选择的课程。

```
CREATE FUNCTION [dbo].[SeltCourse1]
( @sno char(10) )
RETURNS table
AS
RETURN
SELECT Course.课程号, 课程名, 姓名 as 教师姓名,学分,学时 FROM Course,
Teacher where Course.教师号 = Teacher.教师号 and 课程号 not in
(select 课程号 from CS where 学号 = @sno)
```

(18) 表值函数 SeltCourse2,用于实现学生选课时显示该学生已经选择的课程。

```
CREATE FUNCTION [dbo].[SeltCourse2]
( @sno char(10) )
RETURNS table
AS
RETURN
SELECT Course.课程号, 课程名, 姓名 as 教师姓名,学分,学时 FROM Course,
Teacher where Course.教师号 = Teacher.教师号 and 课程号 in
(select 课程号 from CS where 学号 = @sno)
```

为确保数据库安全,创建了 SQL Server 验证的登录账号 user1,作为应用系统连接数据库的账号,密码为"123,,,",并将其映射为 teaching 数据库的用户,将以上视图 1~视图 4 的查询、插入、修改和删除数据的权限授予了此用户,还为其授予了其他视图的查询权限、所有存储过程的执行权限、所有函数的查询权限。

为实现密码列的加密,创建了以下密钥和证书:

(1) 创建数据库主密钥。

```
USE teaching
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'yinQQ-110'
```

(2) 使用数据库主密钥'yinQQ-110'加密创建的证书。

```
CREATE CERTIFICATE cert_Pwd                    -- 创建证书,证书名 cert_Pwd
WITH SUBJECT = 'encrypt AcNo and pwd',         -- 证书的主题
START_DATE = '01/01/2017',                     -- 证书启用日期
EXPIRY_DATE = '01/01/2019'                     -- 证书到期日期
GO
```

(3) 用证书创建对称密钥。

```
CREATE SYMMETRIC KEY PWDKEY
WITH ALGORITHM  = AES_256
ENCRYPTION BY CERTIFICATE cert_Pwd             -- 使用证书 cert_Pwd 加密对称密钥
```

插入或修改学生、教师和管理员的密码时,用对称密钥 PWDKEY 加密保存。

6. 数据库运行、维护

数据库投入运行标志着开发任务的基本完成和维护工作的开始。维护工作是一个长期的过程,包括对数据库设计的评价、调整、修改等工作,这里不再讲述。

14.3.2 应用系统设计与实现

根据需求分析阶段的系统功能,将“教学管理系统”分成 4 个模块,包括管理员管理模块、教师管理模块、学生管理模块和公共模块,如图 14-11 所示。

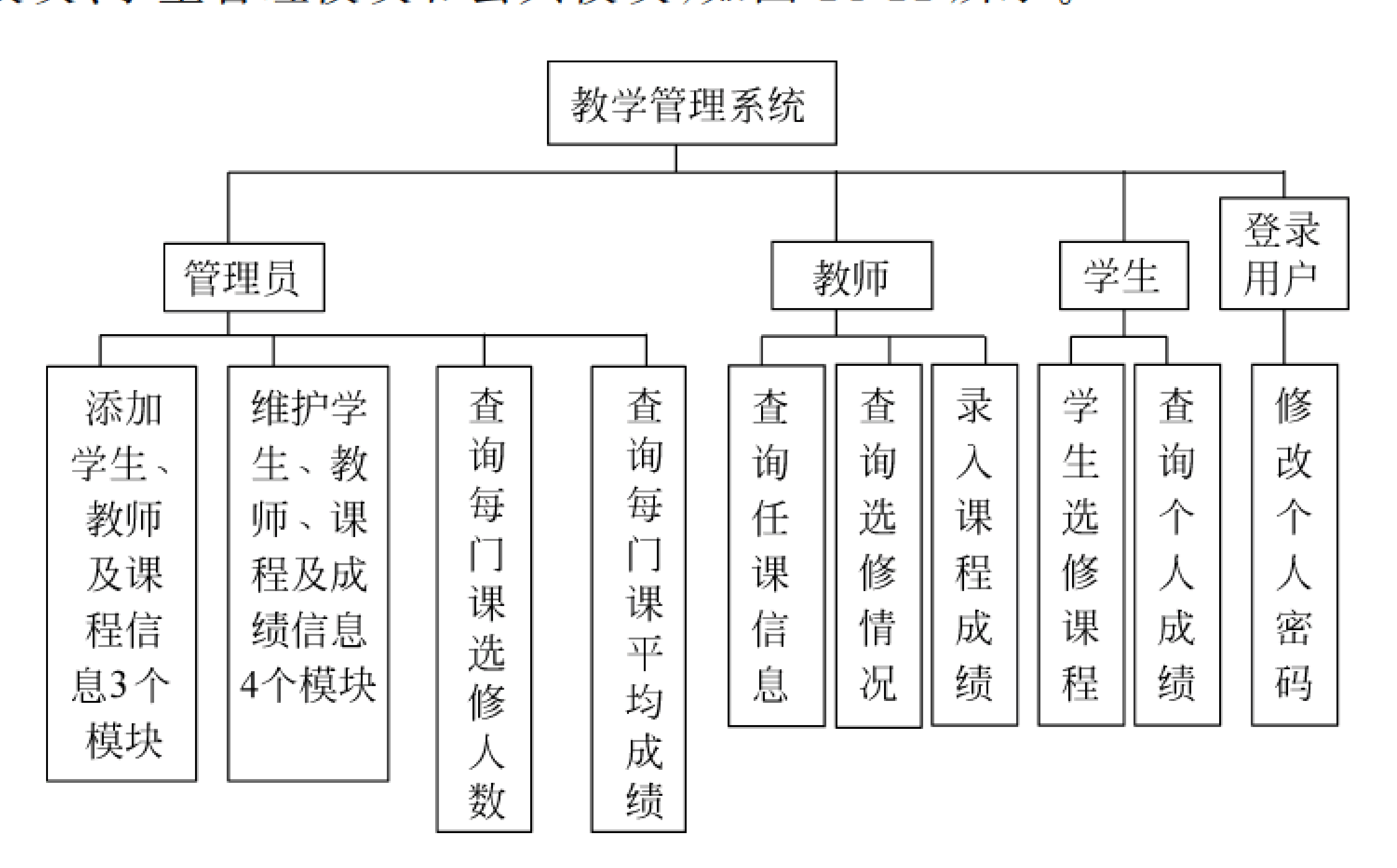


图 14-11　系统功能模块

根据以上 4 个模块,准备设计以下页面:

(1) 登录页:Login. aspx。

(2) 管理员母版页:Site1. Master。

(3) 管理员主页:Main. aspx。

(4) 添加学生页:adminStudentAdd. aspx。

(5) 学生信息维护页:adminStudentDetails. aspx。

(6) 添加课程页:adminCourseAdd. aspx。

(7) 课程信息维护页:adminCourseDetails. aspx。

(8) 添加教师页:adminTeacherAdd. aspx。

(9) 教师信息维护页：adminTeacherDetails. aspx。

(10) 成绩维护页：adminStudentScore. aspx。

(11) 查询每门课选修人数页：adminSearchNumber. aspx。

(12) 查询每门课平均成绩页：adminSearchScore. aspx。

(13) 教师母版页：Site2. Master。

(14) 教师主页：teacher. aspx。

(15) 教师查询任课信息页：teacherCourseDetails. aspx。

(16) 教师查询选课情况页：teacherQueryElect. aspx。

(17) 教师录入成绩页：teacherSubmitScore. aspx。

(18) 学生母版页：Site3. Master。

(19) 学生主页：student. aspx。

(20) 学生选修课程页：studentElect. aspx。

(21) 个人成绩查询页：studentQueryScore. aspx。

(22) 修改个人密码页：ModifyPwd. aspx。

(23) 系统退出页：quit. aspx。

1. 创建项目

(1) 首先启动 Microsoft Visual Studio 2013，进入. NET 的 IDE 界面，准备创建一个新项目。选择“文件”→“新建”→“项目”命令，如图 14-12 所示。



图 14-12 新建项目

(2) 在“新建项目”对话框中选择建立基于 Visual C# 语言的 ASP. NET Web 应用程序，选择空项目模板(Empty)，设置项目文件的名称及存储位置，以及解决方案的名称 teaching_management，如图 14-13 所示。

(3) 单击“确定”按钮，项目建立成功，在窗口右侧“解决方案资源管理器”中可以看到项目目录。下面可以开始在其中创建网页。右击项目名 teaching_management，在弹出的快



图 14-13 新建 ASP. NET Web 应用程序

捷菜单中选择“添加”→“新建项”命令，如图 14-14 所示。

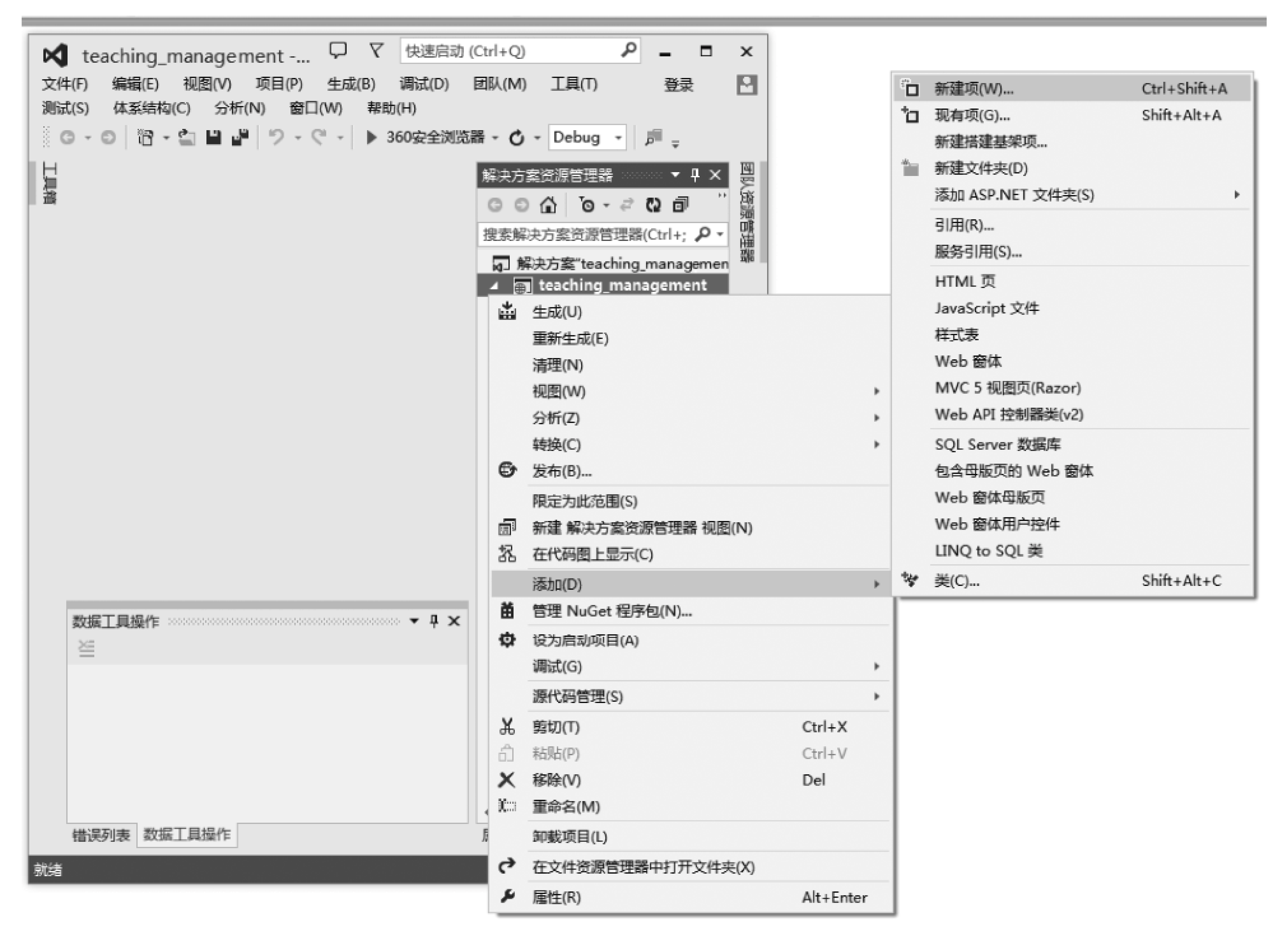


图 14-14 添加新建项

(4) 在“添加新项-teaching_management”窗口选择“Web 窗体”，将其命名为 login.aspx，如图 14-15 所示。

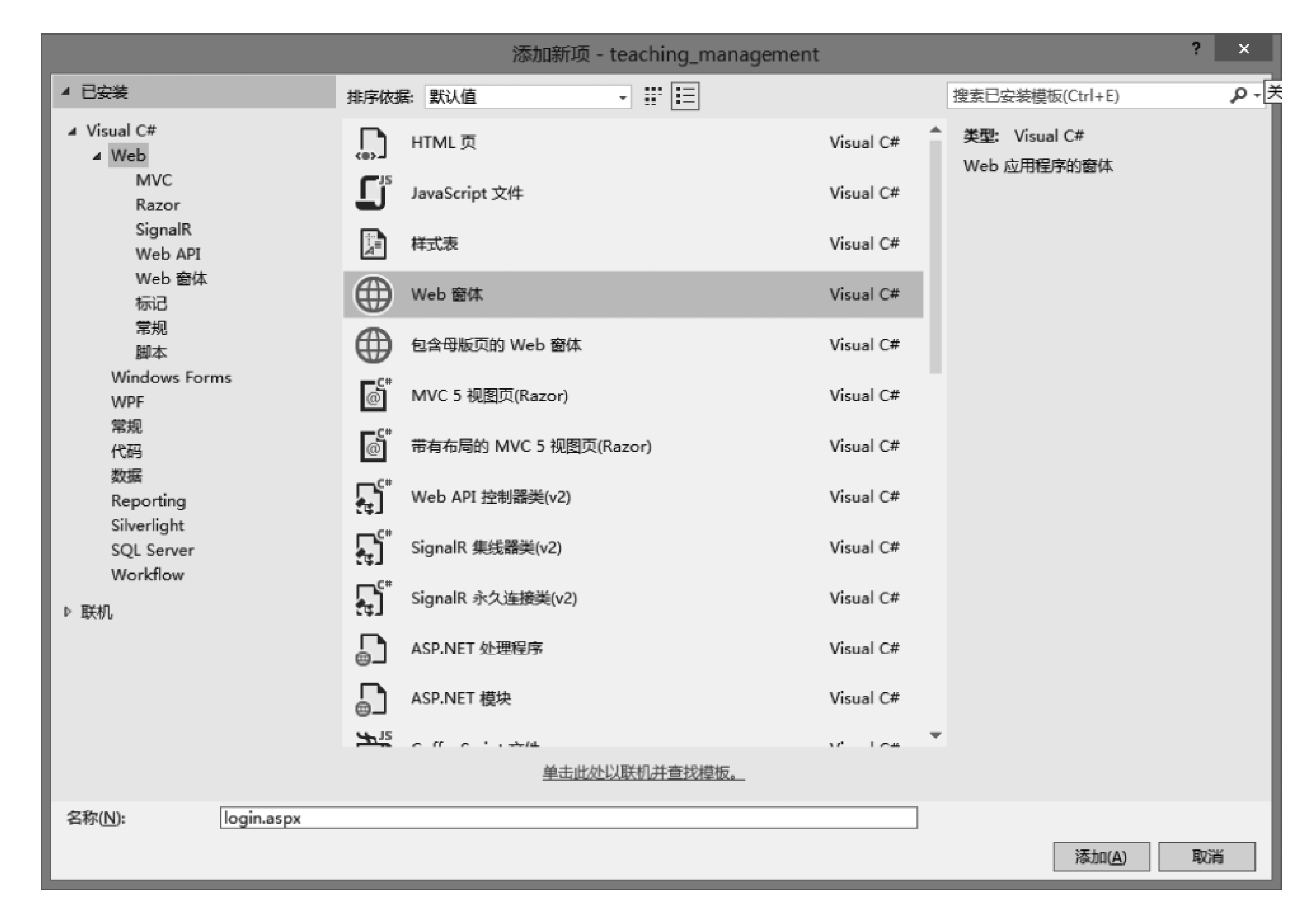


图 14-15　添加 login. aspx 页

(5) login. aspx 页添加成功后，在窗口右侧“解决方案资源管理器”中可以看到此文件名，如图 14-16 所示，下面可以开始进行此网页的前台界面设计和后台代码编写。

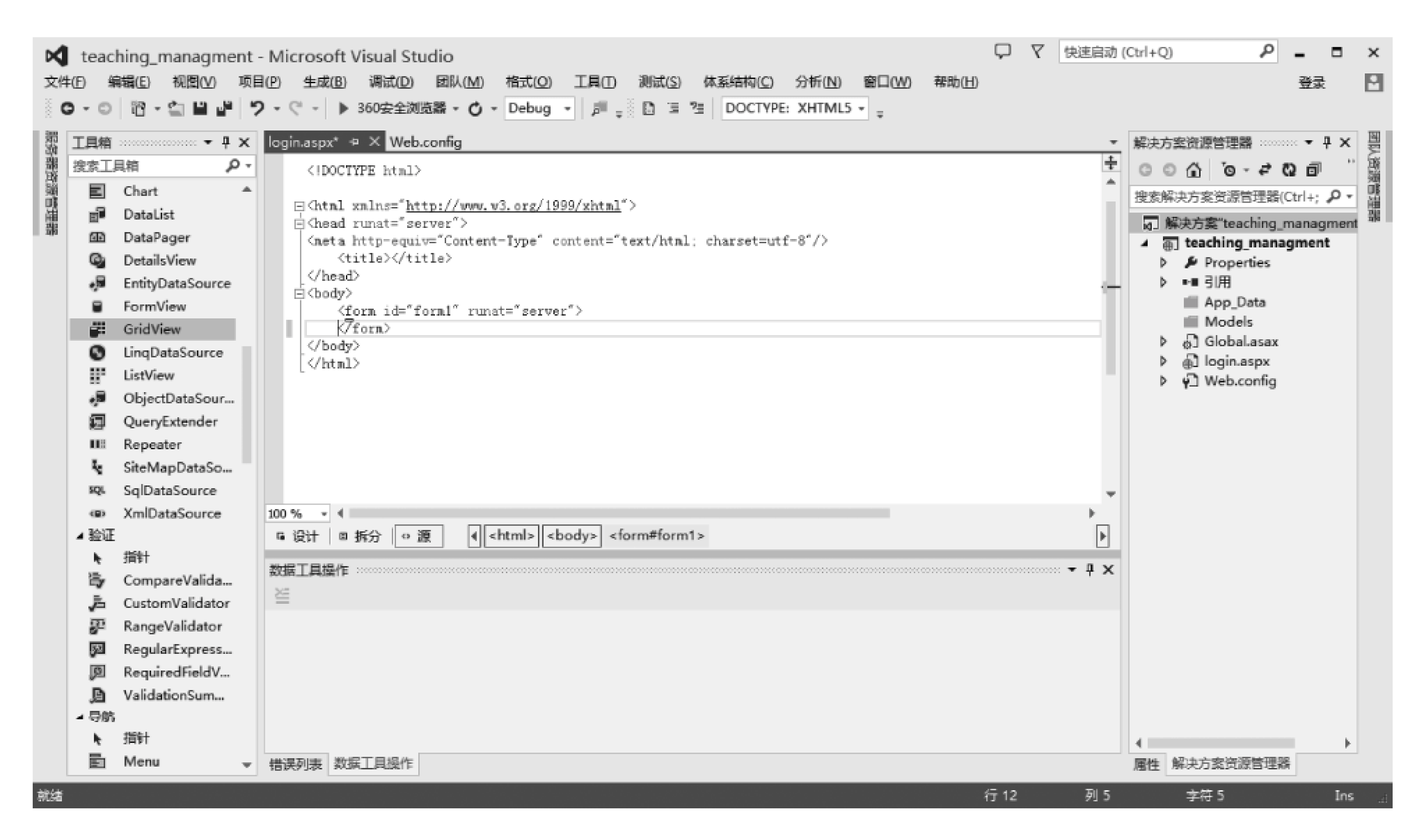


图 14-16　login. aspx 页添加成功

在进行所有的页面设计之前，在 web. config 文件中配置数据源连接，代码如下：

```
<connectionStrings>
  <add name = "teachingConnectionString" connectionString = "Data
  Source = (LOCAL)\YIN;Initial Catalog = teaching;User ID = user1;Password = 123,,,"
  providerName = "System.Data.SqlClient" />
</connectionStrings>
```

2. 设计登录页

登录页面(Login. aspx)是教学管理系统的入口页面，任何未登录的用户都不允许访问本系统的任何信息。教师、学生和管理员用户都可以通过登录页面进入教学管理系统。

登录页面的网页设计前台界面利用表格及各种 ASP. NET 控件(如 TextBox、Label、Button 等)来实现，如图 14-17 所示。



图 14-17　登录页面的前台设计界面

登录页的后台功能代码如下：

```
namespace teaching_management
{
    public partial class Login : System.Web.UI.Page
    {
        string s = System.Configuration.ConfigurationManager.ConnectionStrings
        ["TeachingConnectionString"].ConnectionString;          //连接串变量
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        //登录按钮 Click 事件
        protected void Button1_Click(object sender, EventArgs e)
        {
            SqlConnection conn = new SqlConnection(s);
            string userName = TextBox1.Text.Trim();              /*用户的登录名*/
            string userPwd = TextBox2.Text.Trim();               /*用户的密码*/
            string userRole = RadioButtonList1.SelectedValue;    /*用户的角色*/
            string selectStr = "";
            string pname;
```

```
            switch(userRole)
            {
                case "0" ://角色为教师
                    selectStr = "exec loginTea '" + userName + "','" + userPwd + "'";
                   break;
                case "1"://角色为学生
                   selectStr = "exec loginStu '" + userName + "','" + userPwd + "'";
                   break;
                case "2"://角色为管理员
                   selectStr = "exec loginMan '" + userName + "','" + userPwd + "'";
                     break;
            }
            SqlCommand cmd = new SqlCommand(selectStr,conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            if (dr.Read())
             {
                    Session["UserName"] = userName;
                    Session["UserN"] = (string) dr[0];          //取用户姓名
                    Session["userRole"] = userRole;
                    TextBox1.Text = "";
                    TextBox2.Text = "";
                    Label3.Text = "";
                    switch (userRole)
                    {
                            case "0":
                                 Response.Redirect("teacher.aspx");
                                 break;
                            case "1":
                                 Response.Redirect("student.aspx");
                                 break;
                            case "2":
                                 Response.Redirect("Main.aspx");
                                 break;
                    }
             }
         else Label3.Text = "用户名或密码错误!";
        }
      //重置按钮 Click 事件
        protected void Button2_Click(object sender, EventArgs e)
        {
            TextBox1.Text = "";
            TextBox2.Text = "";
            Label3.Text = "";
            RadioButtonList1.SelectedValue = "1";
        }
    }
}
```

3. 设计管理员母版页

管理员母版页(Site1.Master)为所有管理员操作页面的母版。右击项目名 teaching_

management，在弹出的快捷菜单中选择“添加”→“Web 窗体母版页”命令，如图 14-18 所示。

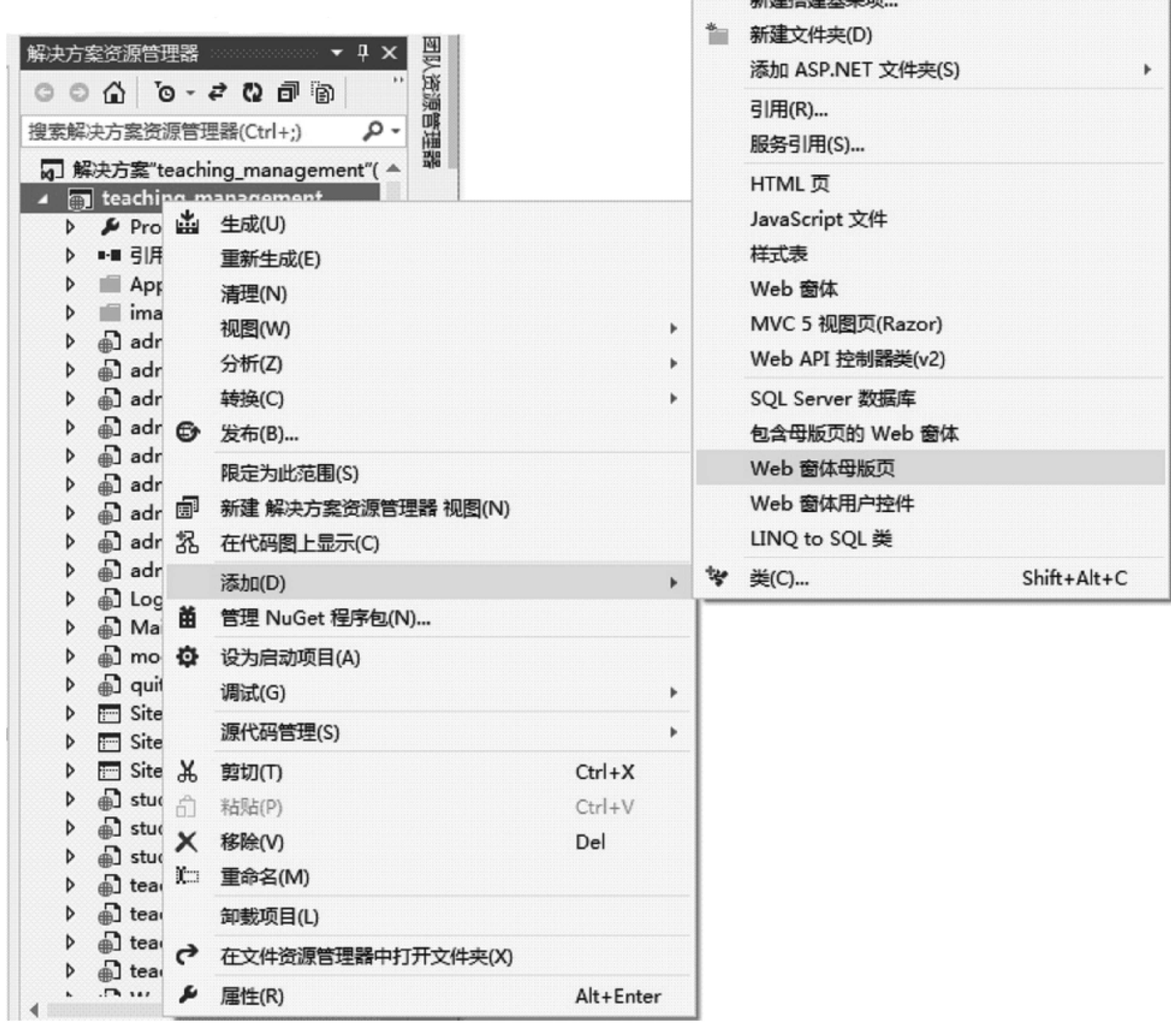


图 14-18 添加 Web 窗体母版页

管理员母版页的网页设计前台界面利用表格及各种 ASP. NET 控件(如 Menu、Label、Button 等)来实现，其中，Menu 可以通过“编辑菜单项”进行子菜单设置，如图 14-19 所示。

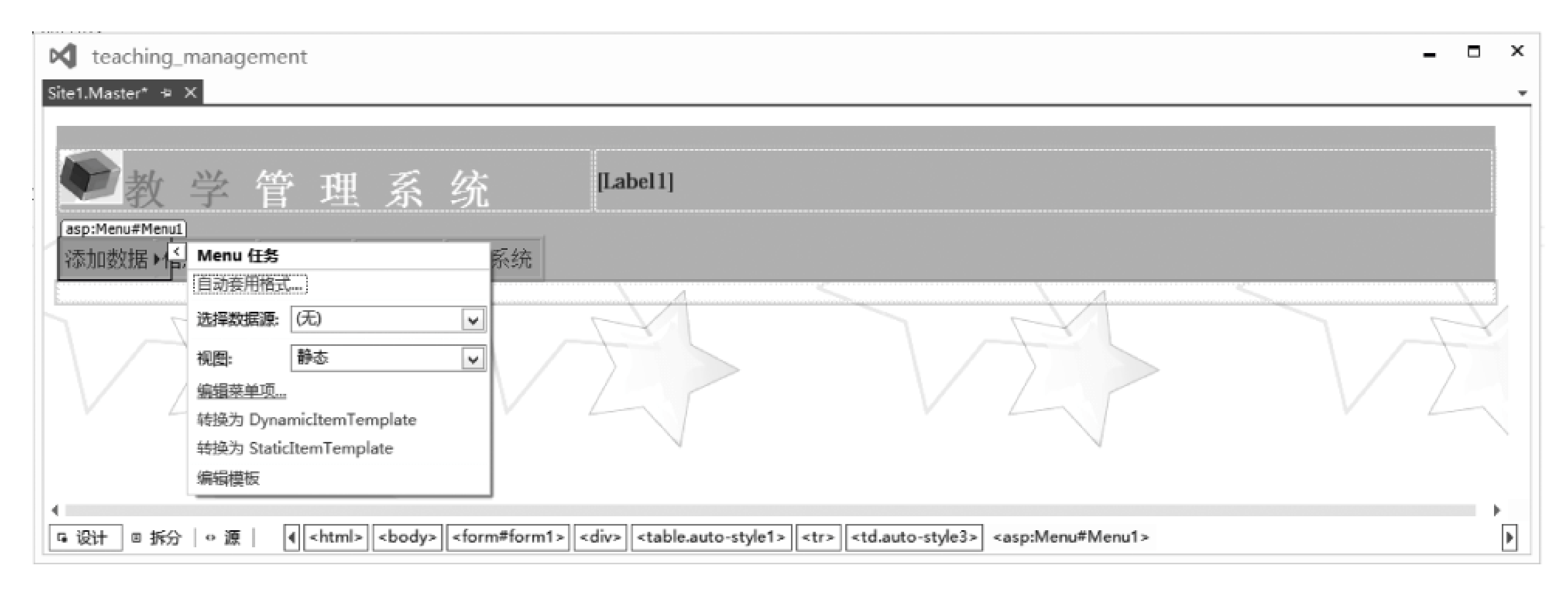


图 14-19 管理员母版页界面

在 Menu 的“菜单项编辑器”窗口可以对各子菜单项的某些属性进行设置。例如，“添加学生”菜单项的文本和链接页面设置，如图 14-20 所示。

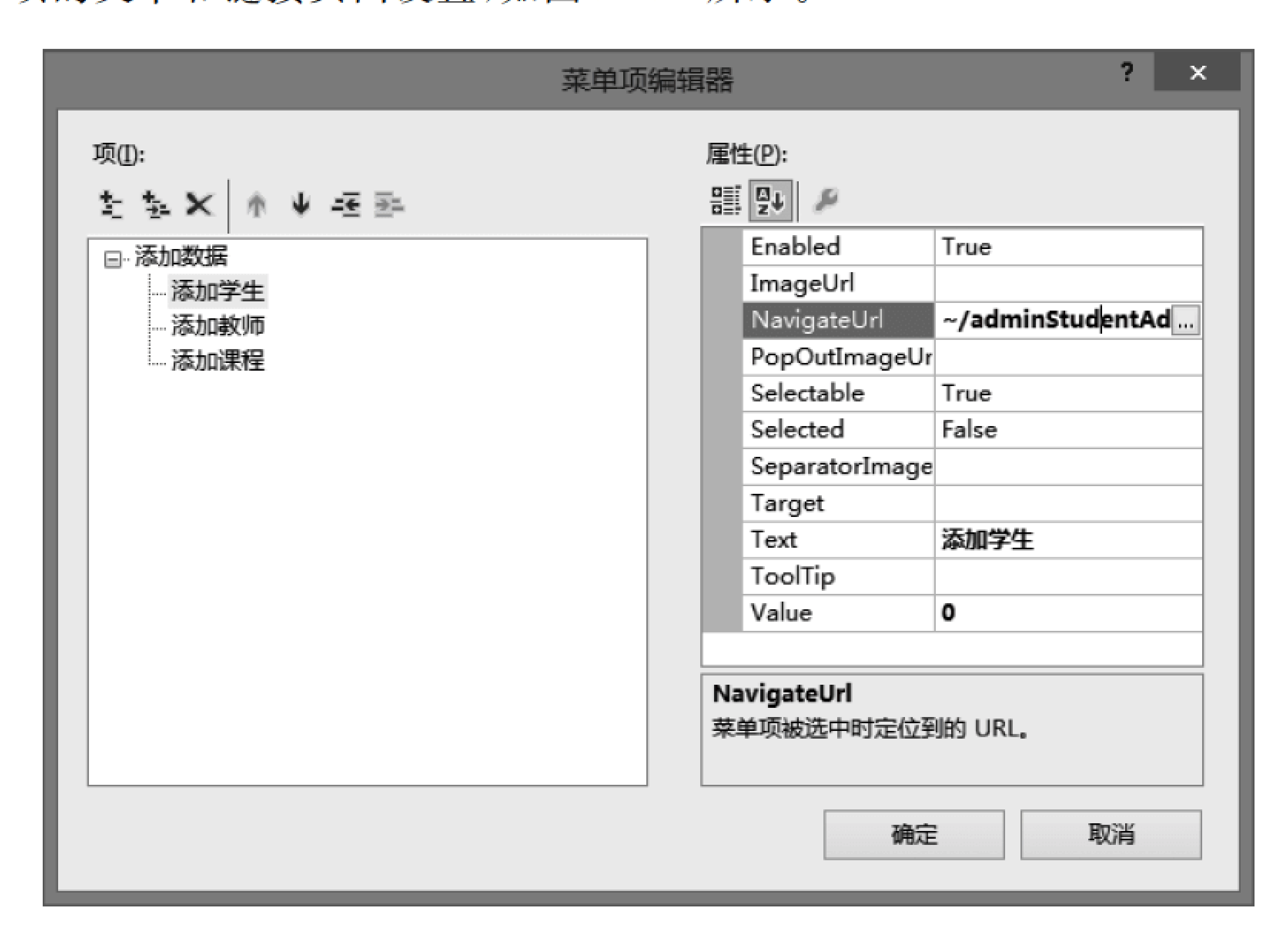


图 14-20 菜单项编辑器

管理员母版页通过后台代码设置只有管理员才能进入，其他用户禁止进入，代码如下：

```
namespace teaching_management
{
    public partial class Site1 : System.Web.UI.MasterPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {   //设置只有管理用户可以进入
            if((string)Session["userRole"]!= "2")
                Response.Redirect("Login.aspx");
            else
                Label1.Text = "欢迎您：" + Session["userN"] + "~~~~~~~";
        }
    }
}
```

4. 设计管理员主页

管理员主页(Main.aspx)的界面和功能与管理员母版页完全相同，右击项目名 teaching_management，在弹出的快捷菜单中选择“添加”→“包含母版页的 Web 窗体”，如图 14-21 所示。

在弹出的“选择母版页”对话框中选择 Site1.Master 母版页，如图 14-22 所示，此页面只是一个导航页，不需要添加任何内容。

5. 设计添加学生页

添加学生页(adminStudentAdd.aspx)在创建时也要包含 Site1.Master 母版页。管理



图 14-21　创建管理员主页

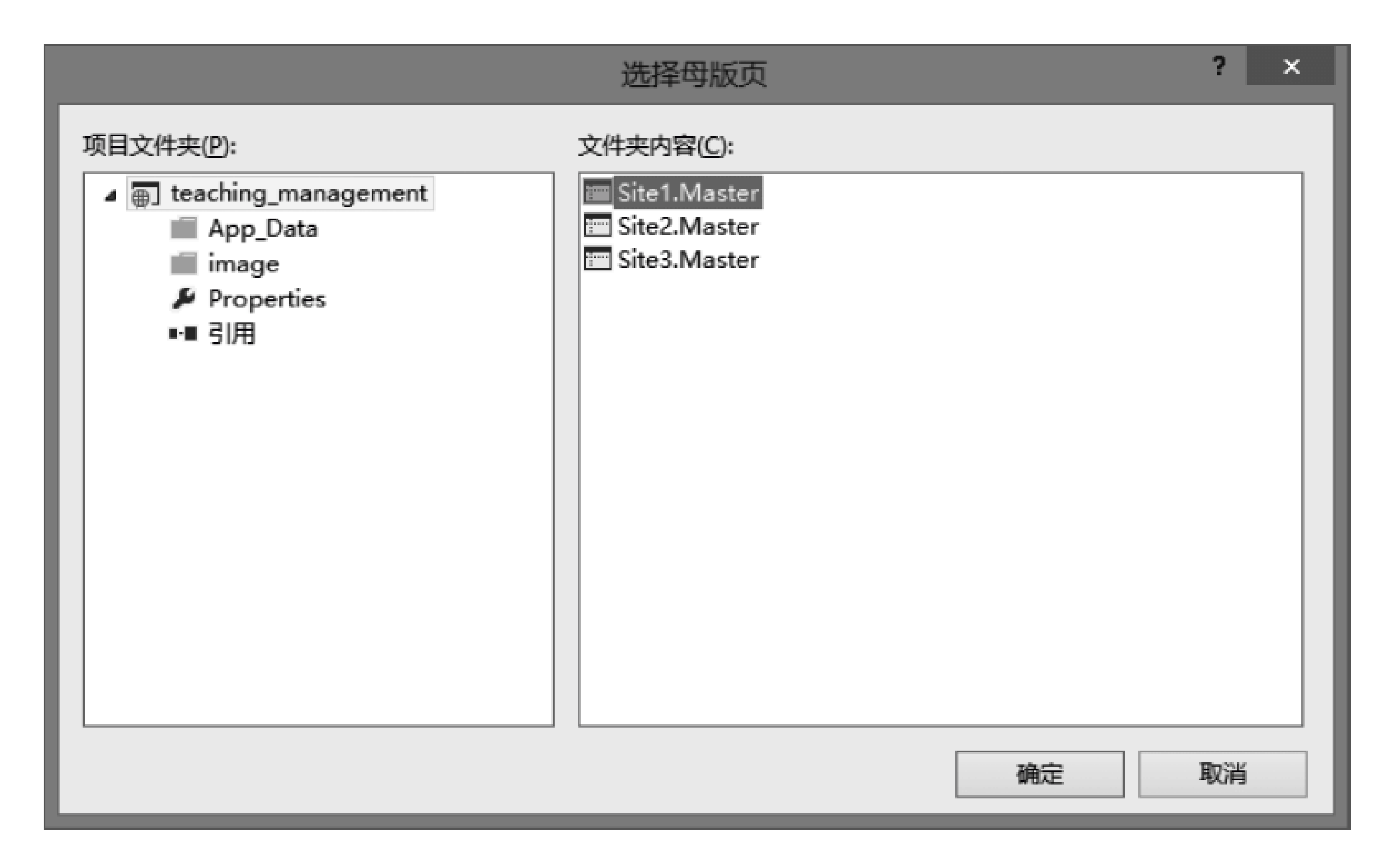


图 14-22　选择母版页

员用户在此页可以完成新学生信息的添加，所以此页面的前台界面需添加表格及各种ASP.NET 控件(如 TextBox、Button、DropDownList 等)，如图 14-23 所示。



图 14-23　添加学生页的前台设计界面

添加学生页的后台功能代码如下：

```
namespace teaching_management
{
    public partial class adminStudentAdd : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
         {
         }
        //添加按钮 Click 事件
        protected void Button1_Click(object sender, EventArgs e)
         {
         string s = System.Configuration.ConfigurationManager.ConnectionStrings
         ["TeachingConnectionString"].ConnectionString;
         SqlConnection conn = new SqlConnection(s);
         string insertStr = "EXEC addstudent '" + TextBox1.Text + "',N'" +
TextBox2.Text + "',N'" + RadioButtonList1.SelectedValue + "',N'" + TextBox3.Text +
"',N'" + DropDownList1.SelectedValue + "',N'" + DropDownList2.SelectedValue + "','" +
TextBox1.Text + "'";
SqlCommand cmd = new SqlCommand(insertStr, conn);
         conn.Open();
         int flag = cmd.ExecuteNonQuery();                         //执行添加
         if (flag > 0)                                             //如果添加成功
             Label2.Text = "成功添加学生信息!";
         else                                                      //如果添加失败
             Label2.Text = "添加学生信息失败,请查看输入是否正确!";
         conn.Close();
         }
         //重置按钮 Click 事件
         protected void Button2_Click(object sender, EventArgs e)
         {
             TextBox1.Text = "";
             TextBox2.Text = "";
```

```
                TextBox3.Text = "";
                RadioButtonList1.SelectedValue = "男";
            }
        }
    }
```

添加成功后的页面运行结果如图 14-24 所示。

教 学 管 理 系 统　欢迎您：张明 ~~~~~~~

添加数据▸　信息维护▸　查询信息▸　修改密码　退出系统

添加学生

成功添加学生信息!

学　号：	2016014405
姓　名：	张宏
性　别：	◉男 ○女
出生日期：	1998-11-12　如：1998-1-11
专　业：	网络工程 ▾
年　级：	2016级 ▾
添加　重置	

图 14-24　成功添加一个学生

6. 设计添加教师页

添加教师页(adminTeacherAdd.aspx)在创建时也要包含 Site1.Master 母版页。管理员用户在此页可以完成新教师信息的添加，所以此页面的前台界面需添加表格及各种 ASP.NET 控件(如 TextBox、Button、DropDownList 等)，其中添加“学院”用 DropDownList 控件，绑定“教师表”中的“学院”列，并过滤掉重复行，如图 14-25 所示。



图 14-25　添加教师页的前台设计界面

添加教师页面的后台功能代码如下：

```
namespace teaching_management
{
    public partial class adminTeacherAdd : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        //添加按钮 Click 事件
        protected void Button1_Click(object sender, EventArgs e)
        {
         string s = System.Configuration.ConfigurationManager.ConnectionStrings
         ["TeachingConnectionString"].ConnectionString;
         SqlConnection conn = new SqlConnection(s);
         string insertStr = "EXEC addteacher '" + TextBox1.Text + "',N'" +
TextBox2.Text + "',N'" + RadioButtonList1.SelectedValue + "',N'" +
DropDownList1.SelectedValue + "','" + TextBox1.Text + "'";
         SqlCommand cmd = new SqlCommand(insertStr, conn);
         conn.Open();
         int flag = cmd.ExecuteNonQuery();
         if (flag > 0)
             Label2.Text = "成功添加教师信息!";
         else
             Label2.Text = "添加教师信息失败,请查看输入是否正确!";
         conn.Close();
        }
        //重置按钮 Click 事件
        protected void Button2_Click(object sender, EventArgs e)
        {
            TextBox1.Text = "";
            TextBox2.Text = "";
            RadioButtonList1.SelectedValue = "男";
        }
    }
}
```

添加成功后的页面运行结果如图 14-26 所示。

7. 设计添加课程页

添加课程页(adminCourseAdd.aspx)在创建时也要包含 Site1.Master 母版页。管理员用户在此页可以完成新课程信息的添加，所以此页面的前台界面需添加表格及各种 ASP.NET 控件(如 TextBox、Button、DropDownList 等)，其中添加“任课教师”用 DropDownList 控件，绑定“教师表”中的“教师号”列，如图 14-27 所示。

添加课程页面的后台功能代码如下：

```
namespace teaching_management
{
    public partial class adminCourseAdd : System.Web.UI.Page
    {
```

教 学 管 理 系 统　欢迎您：张明
添加数据▸ 信息维护▸ 查询信息▸ 修改密码 退出系统
添加教师
成功添加教师信息!
教师号： 20015
姓　名： 郑立峰
性　别： ⦿男 ○女
学　院： 信息技术
添加 重置

图 14-26　成功添加一个教师



图 14-27　添加课程页的前台设计界面

```
protected void Page_Load(object sender, EventArgs e)
{
}
//添加按钮 Click 事件
protected void Button1_Click(object sender, EventArgs e)
{
  string s = System.Configuration.ConfigurationManager.ConnectionStrings
  ["TeachingConnectionString"].ConnectionString;
  SqlConnection conn = new SqlConnection(s);
  string insertStr = "insert into Course_view(课程号,课程名,学分,学时,教师号)
values('" + TextBox1.Text + "',N'" + TextBox2.Text + "'," + DropDownList3.SelectedValue + ","
 + DropDownList1.SelectedValue + ",'" + DropDownList2.SelectedValue + "')";
```

```
            SqlCommand cmd = new SqlCommand(insertStr, conn);
            conn.Open();
            int flag = cmd.ExecuteNonQuery();
            if (flag > 0)
                Label2.Text = "成功添加课程信息!";
            else
                Label2.Text = "添加课程信息失败,请查看输入是否正确!";
            conn.Close();
        }
        //重置按钮 Click 事件
        protected void Button2_Click(object sender, EventArgs e)
        {
            TextBox1.Text = "";
            TextBox2.Text = "";
            DropDownList3.SelectedValue = "1";
        }
    }
}
```

添加成功后的页面运行结果如图 14-28 所示。

图 14-28　添加课程成功后的页面

8. 设计学生信息维护页

学生信息维护页(adminStudentDetails.aspx)在创建时也要包含 Site1.Master 母版页。管理员用户在此页面可以完成学生信息的维护,包括学生信息的修改和删除。

学生信息维护页面直接使用 GridView 控件的自带功能来实现修改和删除。首先添加一个 GridView 控件,然后为其设置数据源为 teaching 数据库中的"学生视图(student_view)",并按"专业"和"年级"升序排序,其设计界面如图 14-29 所示。

学生信息维护网页的运行情况如图 14-30 所示。



图 14-29　学生信息维护页设计界面



图 14-30　学生信息维护页的执行

教师信息维护页(adminTeacherDetails.aspx)和课程信息维护页(adminCourseDetails.aspx)同样在创建时也要包含Site1.Master母版页。

管理员用户在这两个页面可以完成教师信息的维护和课程信息的维护，具体功能包括教师信息和课程信息的修改和删除。这两个页面的实现方式都与学生信息维护页面相似，此处不再赘述。

9. 学生成绩维护页

学生成绩维护页面(adminStudentScore.aspx)在创建时也要包含 Site1.Master 母版页。管理员用户在此页面可以完成学生成绩的维护,包括学生成绩的修改和删除。

学生成绩维护页面直接使用 GridView 控件的自带功能来实现。首先添加一个 GridView 控件,然后为其设置数据源为 teaching 数据库中的"选课视图(CS_view)"。此网页的设计界面如图 14-31 所示。



图 14-31 学生成绩维护页设计界面

学生信息维护网页的编辑状态下,只有"成绩"列可以修改或删除,如图 14-32 所示。



图 14-32 学生成绩维护页的执行状态

10. 查询每门课的选修人数页

查询每门课的选修人数页(adminSearchNumber.aspx)在创建时也要包含 Site1.Master 母版页。管理员用户在此页面可以查询每门课的选修人数,来确定哪些课程适合开课,哪些课程不适合开课。

查询每门课的选修人数页面直接使用 GridView 控件的自带功能来实现。首先添加一个 GridView 控件,然后为其设置数据源为 teaching 数据库中的"选课人数视图(Pnum)",并按"选课人数"升序显示。此网页的设计界面如图 14-33 所示。



图 14-33 查询每门课的选修人数页设计界面

查询每门课的选修人数页面的运行结果如图 14-34 所示。



课程号	课程名	教师号	姓名	学院	选课人数
2016-2017-2-0006	c#开发	12121	张平	信息技术	0
2016-2017-2-0001	数据库技术	20021	张三	信息技术	2
2016-2017-2-0004	操作系统	20021	张三	信息技术	1
2016-2017-2-0005	计算机组成原理	20022	李四	信息技术	1
2016-2017-2-0002	计算机网络	20022	李四	信息技术	4
2016-2017-2-0003	JAVA语言	20023	王五	信息技术	3
2016-1017-1-0008	计算机专业英语	20023	王五	信息技术	2

图 14-34 查询每门课的选修人数页的运行结果

11. 设计查询每门课的平均成绩页

查询每门课的平均成绩页面(adminSearchScore. aspx)在创建时也要包含 Site1. Master 母版页。管理员用户在此页面可以查询每门课的平均成绩,以了解学生考试成绩情况。

查询每门课的平均成绩页面直接使用 GridView 控件的自带功能来实现。首先添加一个 GridView 控件,然后为其设置数据源为 teaching 数据库中的“平均成绩视图(Avg)”,并按“平均成绩”升序显示。此网页的设计界面如图 14-35 所示。



图 14-35　查询每门课的平均成绩页设计界面

查询每门课的平均成绩页面的运行结果如图 14-36 所示。



课程号	课程名	教师号	姓名	学院	平均成绩
2016-1017-1-0008	计算机专业英语	20023	王五	信息技术	89
2016-2017-2-0001	数据库技术	20021	张三	信息技术	
2016-2017-2-0002	计算机网络	20022	李四	信息技术	79
2016-2017-2-0003	JAVA语言	20023	王五	信息技术	73
2016-2017-2-0004	操作系统	20021	张三	信息技术	79
2016-2017-2-0005	计算机组成原理	20022	李四	信息技术	

图 14-36　查询每门课的平均成绩页的运行结果

12. 设计教师母版页和主页

教师母版页(Site2. Master)为所有教师操作页面的母版。其创建方法和管理员母版页

相同，界面如图 14-37 所示。教师主页(teacher.aspx)的界面和功能与教师母版页完全相同，创建方法和管理员主页相同。

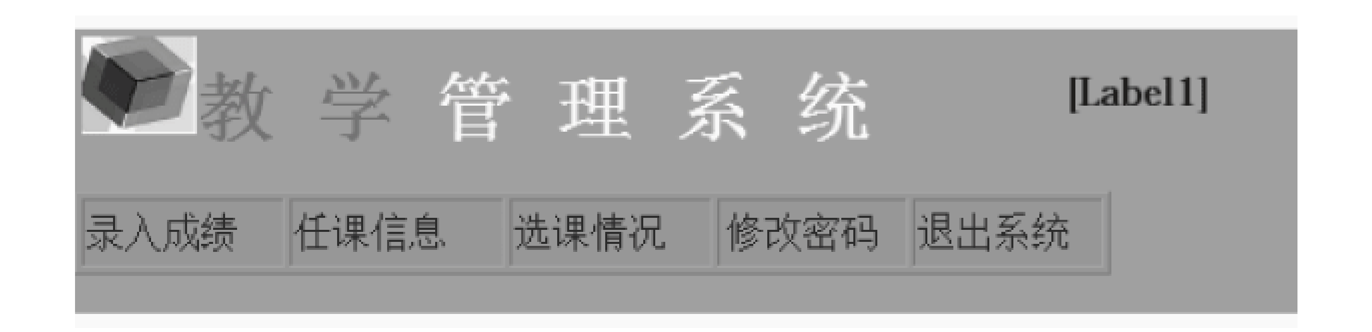


图 14-37　教师母版页前台设计界面

教师母版页通过后台代码设置只有教师才能进入，其他用户禁止进入，代码如下：

```
namespace teaching_management
{
    public partial class Site2 : System.Web.UI.MasterPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {   //设置只有教师用户可以进入
            if ((string)Session["userRole"] != "0")
                Response.Redirect("Login.aspx");
            else
                Label1.Text = "欢迎您: " + Session["userN"] + "~ ~ ~ ~ ~ ~ ~";
        }
    }
}
```

13. 设计教师查询任课信息页

教师查询任课信息页(teacherCourseDetails.aspx)在创建时也要包含 Site2.Master 母版页。教师用户在此页面可以查看自己的任课情况，包括此教师所任课程的详细信息。

教师查询任课信息页面使用 GridView 控件显示进行查询的教师所任课程的详细信息。首先添加一个 GridView 控件，为其设置数据源，查询 teaching 数据库中的“课程视图(course_view)”，查询条件为登录教师的教师号，其设计界面如图 14-38 所示。



图 14-38　教师任课信息页的前台设计界面

教师查询任课信息页面运行结果如图 14-39 所示。



您的任课信息：

课程号	课程名	学分	学时	教师号
2016-2017-2-0001	数据库技术	4	90	20021
2016-2017-2-0004	操作系统	3	54	20021

图 14-39　教师任课信息查询

14. 设计教师查询选课情况页

教师查询选课情况页面(teacherQueryElect.aspx)在创建时也要包含 Site2.Master 母版页。教师用户在此页面可以查看自己所任课程的选修情况,包括每门课程选修的学生的详细信息。

教师查询选课情况页面使用 GridView 控件显示进行查询的教师所任课程的选修情况。首先添加一个下拉列表控件,用于选择要查询的"课程名称",再添加一个 GridView 控件,然后利用后台代码实现查询数据与此控件的绑定,其设计界面如图 14-40 所示。

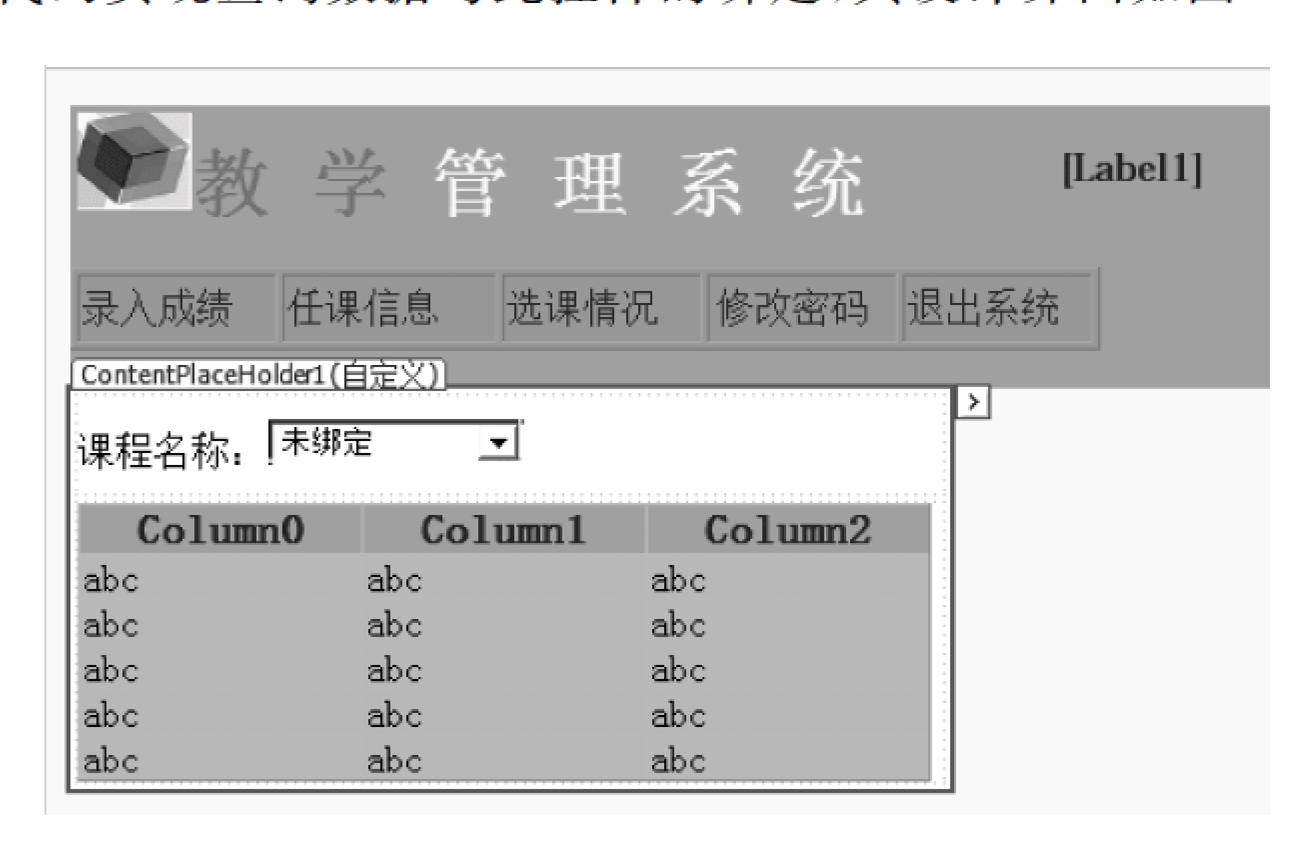


图 14-40　教师查询选课情况页的前台设计界面

教师查询选课情况页面的后台功能代码如下:

```
namespace teaching_management
{
    public partial class teacherQueryElect : System.Web.UI.Page
    {
        string s = System.Configuration.ConfigurationManager.ConnectionStrings
["TeachingConnectionString"].ConnectionString;
        protected void Page_Load(object sender, EventArgs e)
```

```
            {
                if (!this.IsPostBack)
                {  //在 DropDownList 控件上绑定登录教师任课的课程号和课程名
                    SqlConnection conn = new SqlConnection(s);
                    string selectStr = "exec teacher_cNO_Name '" + (string)Session["UserName"]
+ "'";  //执行存储过程 teacher_cNO_Name,查询任课的课程号和课程名
                    DataSet ds = new DataSet();
                    conn.Open();
                    SqlDataAdapter da = new SqlDataAdapter(selectStr, conn);
                    da.Fill(ds);
                    conn.Close();
                    DropDownList1.DataSource = ds.Tables[0].DefaultView;
                    DropDownList1.DataTextField = "课程名";
                    DropDownList1.DataValueField = "课程号";
                    DropDownList1.DataBind();
                    BindGridView();
                }
            }
         //创建方法,在 GridView 控件上绑定数据
        private void BindGridView()
          {
            SqlConnection conn = new SqlConnection(s);
            //执行存储过程 teacherQElect,查询选修 DropDownList 中课程的学生信息
            string selectStr = "exec teacherQElect N'" + DropDownList1.SelectedValue + "'";
            SqlCommand cmd = new SqlCommand(selectStr, conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            GridView1.DataSource = dr;
            GridView1.DataBind();
            conn.Close();
          }
        protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
          { // DropDownList 上选定值发生变化时执行 BindGridView 方法
            BindGridView();
          }
      }
}
```

教师查询选课情况页面运行结果如图 14-41 所示。

15. 设计教师录入成绩页

教师录入成绩页面(teacherSubmitScore.aspx)在创建时也要包含 Site2.Master 母版页。教师用户在此页面可以录入自己所任课程的考试成绩。

教师录入成绩页面使用 GridView 控件显示选修了此教师所任课程的学生的基本信息和“成绩”列,其中成绩列被设置为文本框控件,可以录入和修改,其他列均为只读的 Label 控件。首先添加一个下拉列表控件,用于选择录入成绩的“课程名称”,再添加一个 GridView 控件,然后利用后台代码实现查询数据与此控件的绑定,其设计界面如图 14-42 所示。



学号	姓名	年级	专业
2015014411	张强	2015级	网络工程
2016014402	李华	2016级	网络工程
2015014412	王丽丽	2015级	计算机应用
2016014401	郑晓娜	2016级	软件工程

图 14-41　教师查询选课情况



图 14-42　教师录入成绩页的前台设计界面

教师录入成绩页面的后台功能代码如下：

```
namespace teaching_management
{
    public partial class teacherSubmitScore : System.Web.UI.Page
    {
      string s = System.Configuration.ConfigurationManager.ConnectionStrings
      ["TeachingConnectionString"].ConnectionString;
      protected void Page_Load(object sender, EventArgs e)
      {
         if(!this.IsPostBack)
         { //在 DropDownList 控件上绑定登录教师有学生选修的课程号和课程名
           SqlConnection conn = new SqlConnection(s);
           string selectStr = "exec teacher_cs_NO_Name '" +
                        (string)Session["UserName"] + "'";
           DataSet ds = new DataSet();
           conn.Open();
```

```
            SqlDataAdapter da = new SqlDataAdapter(selectStr, conn);
            da.Fill(ds);
            conn.Close();
            DropDownList1.DataSource = ds.Tables[0].DefaultView;
            DropDownList1.DataTextField = "课程名";
            DropDownList1.DataValueField = "课程号";
            DropDownList1.DataBind();
            BindGridView();
          }
        }
       private void BindGridView()
        {
          SqlConnection conn = new SqlConnection(s);
          //查询选修 DropDownList 中课程的学生基本信息及考试成绩
          string selectStr = "exec teacher_c_score N'" + DropDownList1.SelectedValue + "'";
          SqlCommand cmd = new SqlCommand(selectStr, conn);
          conn.Open();
          SqlDataReader dr = cmd.ExecuteReader();
          GridView1.DataSource = dr;
          GridView1.DataBind();
          conn.Close();
        }
      protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
       {
          BindGridView();
          Label2.Text = "";
       }
      protected void Button1_Click(object sender, EventArgs e)
       {
          SqlConnection conn = new SqlConnection(s);
          SqlCommand cmd = conn.CreateCommand();
          int score;
          string stuID;
          conn.Open();
          for (int i = 1; i < GridView1.Rows.Count; i++)
          {
            score = int.Parse(((TextBox)GridView1.Rows[i].FindControl("TextBox1")).Text.
Trim());
            stuID = GridView1.Rows[i].Cells[0].Text;
            cmd.CommandText = "UPDATE cs_view set 成绩 = " + score + "where 学号 = '" + stuID
+ "' and 课程号 = N'" + DropDownList1.SelectedValue + "'";
           cmd.ExecuteNonQuery();
          }
         conn.Close();
         Label2.Text = "学生成绩录入并提交成功!!";
       }
    }
}
```

教师录入和提交成绩运行结果如图 14-43 所示。

教 学 管 理 系 统　　欢迎您：李四～～～～～～～

录入成绩 任课信息 选课情况 修改密码 退出系统

课程名称：计算机网络

学号	姓名	专业	年级	成绩
2015014411	张强	网络工程	2015级	80
2015014412	王丽丽	计算机应用	2015级	79
2016014401	郑晓娜	软件工程	2016级	90
2016014402	李华	网络工程	2016级	69

提交成绩

学生成绩录入并提交成功！！

图 14-43　教师录入、提交成绩

16. 设计学生母版页和主页

学生母版页(Site3.Master)为所有学生操作页面的母版。其创建方法和管理员母版页相同,界面如图 14-44 所示。学生主页(student.aspx)的界面和功能与学生母版页完全相同,创建方法和管理员主页相同。

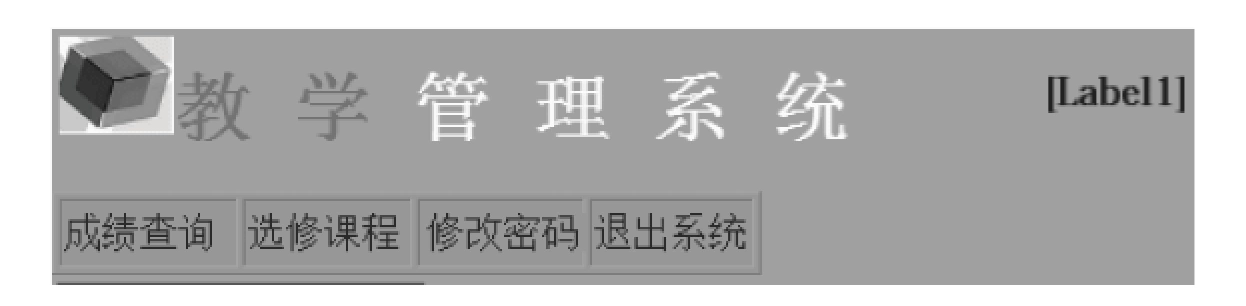


图 14-44　学生母版页前台设计界面

学生母版页通过后台代码设置只有学生才能进入,其他用户禁止进入,代码如下：

```
namespace teaching_management
{
    public partial class Site3 : System.Web.UI.MasterPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {   //设置只有学生用户可以进入
            if ((string)Session["userRole"] != "1")
                Response.Redirect("Login.aspx");
            else
                Label1.Text = "欢迎您: " + Session["userN"] + "～～～～～～～";
        }
    }
}
```

17. 设计学生选修课程页

学生选修课程页面(studentElect.aspx)在创建时也要包含 Site3.Master 母版页。学生用户在此页面可以选修自己想选的课程和退选已选择的课程。

学生选修课程页面采用GridView控件显示所有此学生可选的课程和已选的课程。首先添加两个GridView控件，分别为它们设置可选的课程和已选的课程作为数据源，并分别在每个GridView控件的"编辑列"选项，加入一个ButtonField按钮，为GridView1控件命名为Selt，文本显示为"选修"，如图14-45所示，GridView2控件命名为Tui，文本显示为"退选"，然后利用后台代码实现选修和退选功能。

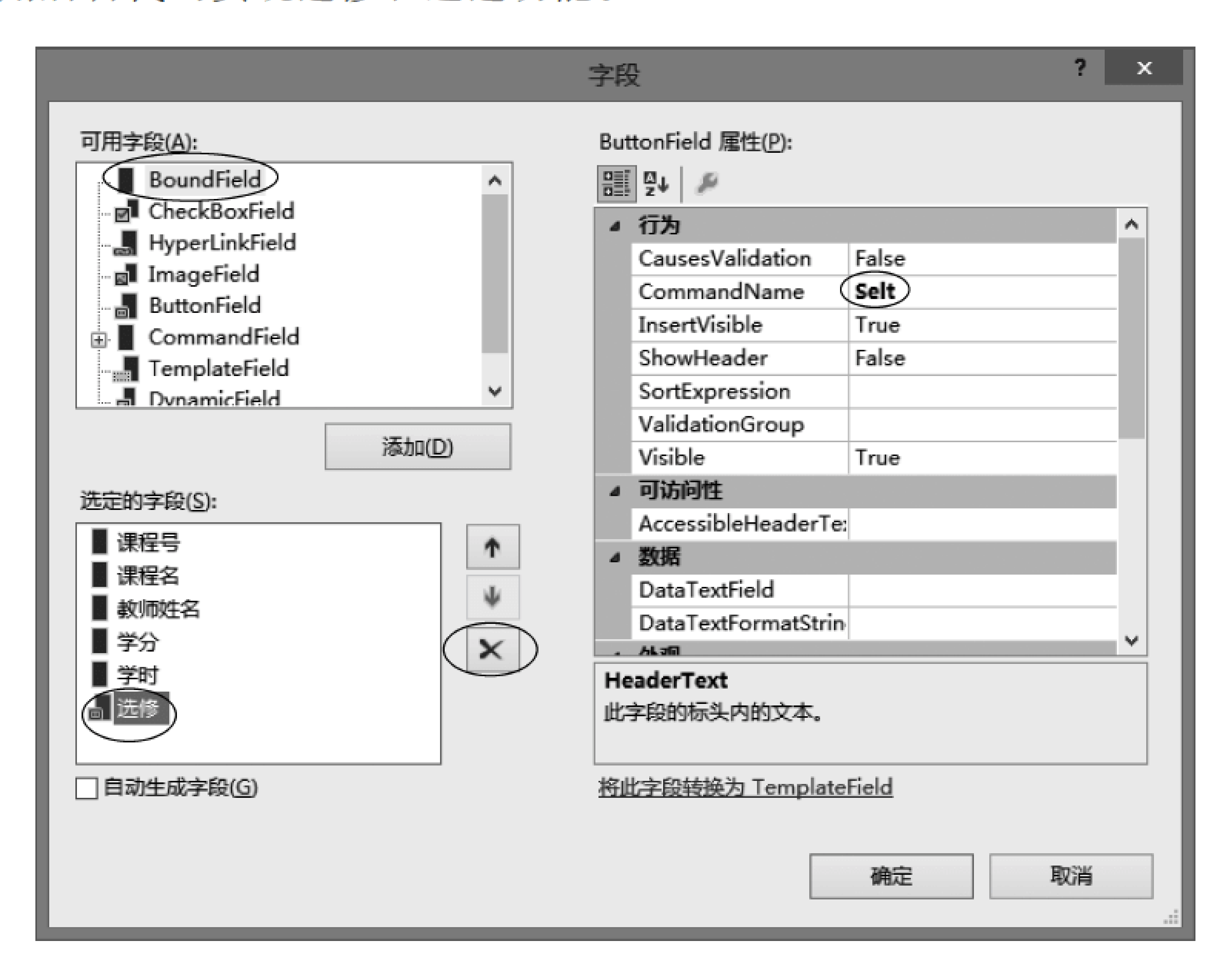


图 14-45 GridView 控件的"选修"按钮设计

学生选修课程页的设计界面如图14-46所示。



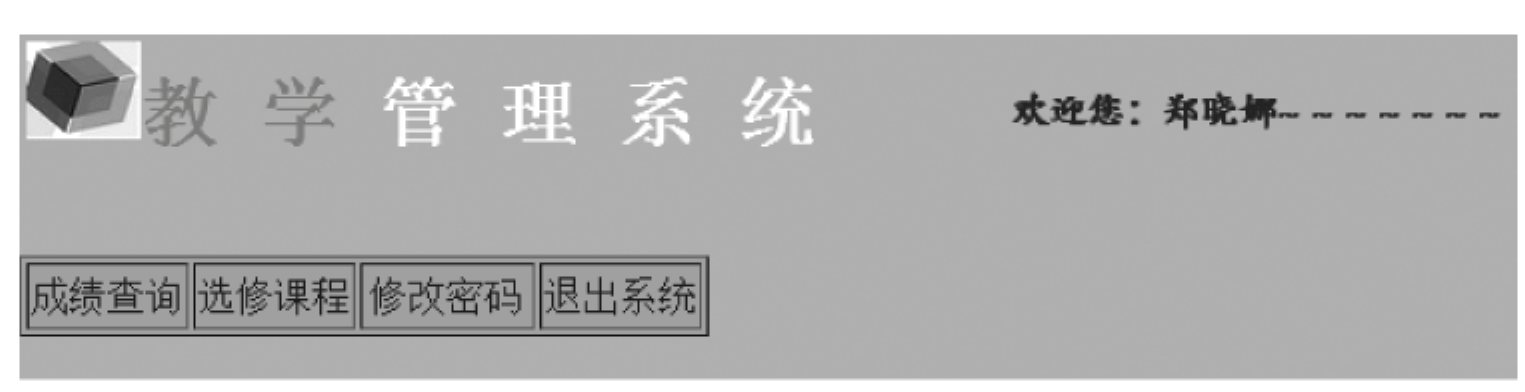
图 14-46 学生选修课程页的前台设计界面

学生选修课程页面的后台功能代码如下：

```
namespace teaching_management
{
  public partial class studentElect : System.Web.UI.Page
   {
    string s = System.Configuration.ConfigurationManager.ConnectionStrings
    ["TeachingConnectionString"].ConnectionString;
    protected void Page_Load(object sender, EventArgs e)
      {
         BindGridView1();
         BindGridView2();
       }
     //GridView1 控件的 RowCommand 事件
     protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
      {
         if (e.CommandName == "Selt")              //如果单击的是"选修"按钮
         {  //取出选修课程所在的行索引
            int index1 = Convert.ToInt32(e.CommandArgument);
            //取出课程号主码值
            string CourseID = GridView1.Rows[index1].Cells[0].Text.ToString();
            SqlConnection conn = new SqlConnection(s);
            string insertStr = "insert into cs_view (学号,课程号)values('" + (string)
Session["UserName"] + "',N'" + CourseID + "')";   //插入选修课程
            conn.Open();
            SqlCommand cmd = new SqlCommand(insertStr,conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            BindGridView1();
            BindGridView2();
         }
      }
   //GridView2 控件的 RowCommand 事件
   protected void GridView2_RowCommand(object sender, GridViewCommandEventArgs e)
     {
        if (e.CommandName == "Tui")               //如果单击的是"退选"按钮
         {
            int index1 = Convert.ToInt32(e.CommandArgument);
            string CourseID = GridView2.Rows[index1].Cells[0].Text.ToString();
            SqlConnection conn = new SqlConnection(s);
            string deleteStr = "delete from cs_view where 学号 = '" + (string)Session
["UserName"] + "'and 课程号 = N'" + CourseID + "'"; //删除选修课程
            conn.Open();
            SqlCommand cmd = new SqlCommand(deleteStr, conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            BindGridView2();
            BindGridView1();
         }
     }
     private void BindGridView1()
```

```
        {
          SqlConnection conn = new SqlConnection(s);
          string selectStr = "SELECT * FROM SeltCourse1(N'" + (string)Session["UserName"] +
"')";  //调用函数 SeltCourse1,查询登录的学生未选修的课程信息
          conn.Open();
          SqlCommand cmd = new SqlCommand(selectStr, conn);
          SqlDataReader dr = cmd.ExecuteReader();
          GridView1.DataSource = dr;
          GridView1.DataBind();
          conn.Close();
         }
        private void BindGridView2()
         {
          SqlConnection conn = new SqlConnection(s);
          string selectStr = "SELECT * FROM SeltCourse2(N'" + (string)Session["UserName"] +
"')";  //调用函数 SeltCourse2,查询登录的学生已选修的课程信息
          conn.Open();
          SqlCommand cmd = new SqlCommand(selectStr, conn);
          SqlDataReader dr = cmd.ExecuteReader();
          GridView2.DataSource = dr;
          GridView2.DataBind();
          conn.Close();
         }
      }
}
```

学生选修课程运行结果如图 14-47 所示。



您可选修的课程

课程号	课程名	教师姓名	学分	学时	选修
2016-2017-2-0004	操作系统	张三	3	54	选修
2016-2017-2-0005	计算机组成原理	李四	4	90	选修
2016-2017-2-0006	c#开发	张平	3	54	选修

您已选修的课程

课程号	课程名	教师姓名	学分	学时	退选
2016-1017-1-0008	计算机专业英语	王五	1	2	退选
2016-2017-2-0001	数据库技术	张三	4	90	退选
2016-2017-2-0002	计算机网络	李四	3	54	退选
2016-2017-2-0003	JAVA语言	王五	4	72	退选

图 14-47　学生选修课程

18. 设计学生个人成绩查询页

学生个人成绩查询页面(studentQueryScore.aspx)在创建时也要包含 Site3.Master 母版页。学生用户在此页面可以查看自己所选每门课程的考试成绩。

学生个人成绩查询页面采用 GridView 控件显示进行查询的学生的考试成绩情况。添加一个 GridView 控件,然后利用后台代码实现查询数据与此控件的绑定,其设计界面如图 14-48 所示。

图 14-48 学生个人成绩查询页的前台设计界面

学生个人成绩查询页的后台功能代码如下:

```
namespace teaching_management
{
    public partial class studentQueryScore : System.Web.UI.Page
    {
      string s = System.Configuration.ConfigurationManager.ConnectionStrings
["TeachingConnectionString"].ConnectionString;
      protected void Page_Load(object sender, EventArgs e)
      {
        SqlConnection conn = new SqlConnection(s);
        //调用函数 stuscore,查询登录的学生已选修的课程的成绩信息
        string SqlStr = "select * from stuscore(" + "'" + (string)Session["UserName"] + "')";
        conn.Open();
        SqlCommand cmd = new SqlCommand(SqlStr, conn);
        SqlDataReader dr = cmd.ExecuteReader();
        GridView1.DataSource = dr;
        GridView1.DataBind();
        conn.Close();
       }
    }
}
```

学生个人成绩查询页运行结果如图 14-49 所示。

19. 设计修改密码页

修改密码页面(ModifyPwd.aspx)所有登录用户都可以进入,登录用户在此页面可以修改自己的登录密码。



课程号	课程名	成绩
2016-2017-2-0002	计算机网络	80
2016-2017-2-0003	JAVA语言	73
2016-2017-2-0004	操作系统	79
2016-2017-2-0005	计算机组成原理	

图 14-49　学生个人成绩查询结果

修改密码页面设计前台界面利用表格及各种 ASP.NET 控件(如 TextBox、Button 等)来实现,如图 14-50 所示。

图 14-50　修改密码页前台界面

修改密码页的后台功能代码如下：

```
namespace teaching_management
{
    public partial class modifyPwd : System.Web.UI.Page
    {
      string s = System.Configuration.ConfigurationManager.ConnectionStrings
["TeachingConnectionString"].ConnectionString;
      protected void Page_Load(object sender, EventArgs e)
      {
      }  //确定按钮的 Click 事件
      protected void Button1_Click(object sender, EventArgs e)
      {
         if (TextBox2.Text != TextBox3.Text)
             Label1.Text = "两次输入不一致!";
         else
         {
             string username = Session["userName"].ToString();
             string selectStr = "";
```

```
        string updateStr = "";
        switch (Session["userRole"].ToString())
         {
            case "0":   //身份为教师
            selectStr = "EXEC teaNo_Pwd '" + username + "','" + TextBox1.Text.Trim() + "'";
            updateStr = "EXEC modiTeaPwd '" + TextBox2.Text + "','" + username + "'";

            break;
            case"1":   //身份为学生
            selectStr = "EXEC stuNo_Pwd '" + username + "','" + TextBox1.Text.Trim() + "'";
            updateStr = "EXEC modiStuPwd '" + TextBox2.Text + "','" + username + "'";

            break;
            case"2":   //身份为管理员
         selectStr = "EXEC maNo_Pwd '" + username + "','" + TextBox1.Text.Trim() + "'";
         updateStr = "EXEC modiManPwd '" + TextBox2.Text + "','" + username + "'";
       break;
         }
         SqlConnection conn = new SqlConnection(s);
         SqlCommand cmd = new SqlCommand(selectStr, conn);
         conn.Open();
         SqlDataReader dr = cmd.ExecuteReader();
         if(dr.Read())  //如果用户存在且输入的密码正确,修改密码
         {
             dr.Close();
             SqlCommand updateCmd = new SqlCommand(updateStr, conn);
             int i = updateCmd.ExecuteNonQuery();
             if(i>0)
              {
                 Label2.Text = "成功修改密码!";
              }
             else
              {
                 Label2.Text = "修改密码失败!";
              }
         }
         else
         {
             Label2.Text = "您输入的旧密码错误,检查后重新输入!";
         }
        conn.Close();
    }
}
//重置按钮的 Click 事件
protected void Button2_Click(object sender, EventArgs e)
{
    TextBox1.Text = "";
    TextBox2.Text = "";
    TextBox3.Text = "";
    Label1.Text = "";
}
```

```
            //返回按钮的 Click 事件
            protected void Button3_Click(object sender, EventArgs e)
            {
               switch (Session["userRole"].ToString())
               {
                   case "0":   //身份为教师
                       Response.Redirect("teacher.aspx");
                       break;
                   case "1":   //身份为学生
                       Response.Redirect("student.aspx");
                       break;
                   case "2":   //身份为管理员
                       Response.Redirect("Main.aspx");
                       break;
                }
            }
        }
    }
```

20. 设计退出系统页

所有登录用户都可以通过“退出系统”菜单执行退出系统页(quit.aspx)中的代码，实现系统退出。

后台代码如下：

```
namespace teaching_management
{
    public partial class quit : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Session["userRole"] = "";
            Session["userN"] = "";
            Response.Redirect("Login.aspx");
        }
    }
}
```

习　题　14

1. 简述 C# 语言的特点。

2. 简述 ADO.NET 对象模型的 5 个主要组件。

3. 简述 ADO.NET 访问数据库的两种方法。

4. 使用 C# 语言，通过 ASP.NET 技术，创建一个“仓库库存管理系统”的数据库应用系统，包括进货录入、产品库存查询、产品库存维护、出货等功能。数据库的设计过程和程序开发请读者自行完成。

参 考 文 献

[1] 尹志宇，郭晴，等. 数据库原理与应用教程——SQL Server[M].2版.北京：清华大学出版社，2016.
[2] 尹志宇，郭晴，等. 数据库原理与应用教程——SQL Server 2008[M].2版.北京：清华大学出版社，2017.
[3] 秦婧. SQL Server 2012王者归来——基础、安全、开发及性能优化[M]. 北京：清华大学出版社，2014.
[4] 李春葆，金昌，曾平葆. 数据库原理与应用——基于SQL Server 2005[M]. 北京：清华大学出版社，2009.
[5] 于富强，解春燕，等. ASP.NET 4.0网站开发使用教程[M]. 北京：北京大学出版社，2012.

图 书 资 源 支 持

感谢您一直以来对清华版图书的支持和爱护。为了配合本书的使用，本书提供配套的资源，有需求的读者请扫描下方的“书圈”微信公众号二维码，在图书专区下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本书的过程中遇到了什么问题，或者有相关图书出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。

我们的联系方式：

地　　址：北京海淀区双清路学研大厦 A 座 707

邮　　编：100084

电　　话：010－62770175－4604

资源下载：http://www.tup.com.cn

电子邮件：weijj@tup.tsinghua.edu.cn

QQ：883604（请写明您的单位和姓名）



书圈

用微信扫一扫右边的二维码，即可关注清华大学出版社公众号“书圈”。